Bibliographie.

[SWAN96]
M. Swanson, L. Stoller
« Low Latency Workstation Cluster Comunications Using Send-Based Protocols »
UUCS-96-001
Departement of Computer Science, University of Utah, January 1996, 8 pages.

[SWAN95]
J. Carter, A. Davis, R. Kuramkote, C.C. Kuo, L. Stoller, M. Swanson.
« Avalance : A Communication and Memory Architecture for Scalable Parallele Computing »
Proceedings of the 22th International Symposium on Computer Architecture 1995, 21 pages.

[T3D] Anonymous
« Cray T3D Architecture »
http://www.cray.com, 43 pages.

[THOMPSON]
M.D. May, R.M. Shepherd, P.W. Thompson
« Networks, Routers and Transputers : Function, Performances and applications »
IOS Press 1993.

[ZERR93]
B. Zerrouk
« Du routage par intervalles »
Actes de Congrès Biennal de l'AFCET sur les Technologies de l'information, Versailles, 1993.

[ZERR94]
B. Zerrouk, J.M. Blin and A. Greiner
« Encapsulating Networks and Routing »
International Parallele Processing Symposium 1994, 8 pages.

[ZERR96] B. Zerrouk, V. Reibaldi, F. Potter, A. Greiner et A. Derieux
« Un routeur de messages adaptatif »
Actes du 4ème Symposium sur les Architectures Nouvelles de Machines 1996, 10 pages.

[PACHE] P. Pacheco
« A User's Guide to MPI »
University of San Francisco. 1995, 60 pages.

[PATTER94]
Thomas E. Anderson, David E. Culler, David A. Patterson
« A case for NOW »
http://now.cs.berkeley.edu, December 1994, 20 pages.

[PATTER95]
K. K. Keeton, T.E. Anderson, D.A. Patterson.
« LogP Quantified : The Case for Low-Overhead Local Area Networks »
Hot Interconnects 1995, 16 pages.

[PCI] PCI Special Group of Interest
« The PCI Local Bus Specifications. Revision 2.1 »
PCI-SIG, July 1995, 282 pages.

[REIBLADI] V. Reibaldi
« Conception et réalisation d'un routeur de paquets à hautes performances »
Thèse de Doctorat de l'Université Paris VI.
MASI, IBP

[RFC1597]
Y. Rekhter, R. Moskowitz, D. Karrenberg, G. de Groot
« Address Allocation for Private Internets »
Mars 1994. 8 pages.

[SEITZ93] C.L. Seitz, N.J. Boden, J. Seizovic, W.K. Su
« The Design of the Caltech Mosaic C Multicomputer »
Proceedings of the University of Washington Symposium on Integrated Systems, PP 1-22, MIT Press, 1993, 22 pages.

[SEITZ95] N.J. Boden, D. Cohen, R.E. Felderman, A.E. Kulawik, C.L. Seitz, J.N. Seizovic, W.K. Su.
« Myrinet—A Gigabit-per-Second Local-Area Network »
IEEE Micro Feb 1995, 16 pages.

[STUNKEL] C.B. Stunkel, D.G. Shea, B. Abali, M.G. Atkins, C.A. Bender, D.G. Grive, P. Hockschild, D.J. Joseph, B.J. Nathanson, R.A. Swetz, R.F. Stucke, M. Tsao, P.R. Valker
« The SP2 High Performance Switch »
IBM Journal Vol 34 N° 2, 1995, pages 185-204.

[SWAN94]
M. Swanson, L. Stoller
« PPE-level Protocols for Carpet Clusters »
UUCS-94-013
Departement of Computer Science, University of Utah, April 1994, 21 pages.

Bibliographie.

[KATE] M. Katevenis
« Telegraphos : High-Speed Communication Architecture for Parallel and Distributed Computer Systems »
FORTH-ICS / TR 123, May 1994, 37 pages.

[KARAM] V. Karamcheti, A. Chien
« A Comparison of Architectural Support of Messaging in the TMC CM-5 and the Cray T3D »
Proceedings of the 22th International Symposium on Computer Architecture, Seattle, 1995, pages 298-307.

[KERMANI] P. Kermani, L. Kleinrock
« Virtual Cut Through : A new computer communication switching technique »
Computer Networks, Vol 3, 1979, pages 267-286

[LEEUWEN87]
J. Leeuwen, R.B. Tan
« Interval Routing »
The computer Journal 30, 1987, pages 298-307.

[LEEUWEN91]
E.M. Bakker, J. Leeuwen, R.B. Tan
« Linear Inteval routing »
Algorithms Review 2, 1991, pages 45-61.

[LEISER] C. Leiserson, Z. Abuhamdeh, D. Douglas, C. Feynman, M. Ganmukhi, J. Hill, W. Hillis, B. Kuszmaul, M. St Pierre, D. Wells, M. Wong, S-W. Yang, R. Zak.
« The Network Architecture of the Connection Machine CM-5 »
TMC Juillet 1992, 13 pages.

[LENOSKI] D. Lenoski at al.
« The Directory Based Cache Coherence Protocol for the Dash Multiprocessor »
Proceedings of the 17th International Symposium on Computer Architecture, Seattle, Mai 1990, pages 148-159.

[LIN] M. Lin, R. Tsang, D. Du
« Performance Evaluation of the CM-5 Interconnection Network »
IEEE CompCon, Février 1993, 22 pages.

[MARTIN] R. Martin
« HPAM : An active mesage layer for Network of HP Workstation »
Hot Interconnects II, Août 1994.

[MPI] Message Passing Interface Forum
« MPI : A Message-Passing Interface Standard »
International Journal of Supercomputer Applications, vol 8 nos 3 / 4, 1994.

Bibliographie.

[DALLY92] W. Dally, J.A. Fiske, J. Keen; R. Lethin, M. Noakes, P. Nuth
« The Message-Driven Processor »
IEEE Micro, Avril 1992, pages 23-39.

[EICKEN] T. Von Eicken, D.E. Culler, S.C. Goldstein, K.E. Schauser
« Active Messages, A mechanism for Integrated Communication and Computation »
Proceedings of the 19th International Symposium on Computer Architecture, 1992, pages 256-266.

[FELDER] R. Felderman, A. DeSchon, D. Cohen, G. Finn
« Atomic : A High Speed Local Communicationn Architecture »
Journal of High Speed Networks, 1994, pages 1-28.

[HALILA] M. Halila
« Un réseau de communication hautes performances pour systèmes multiprocesseurs à mémoire distribuée ».
Thèse de Doctorat de l'Université Paris VI.
Bull / MASI

[HE&Al-1] Mark Horowitz, A. Gupta, M Rosenblum, J. Hennessy and the FLASH Team
« The Stanford FLASH Multiprocessor »
Proceedings of the 21th International Symposium on Computer Architecture. 1994, pages 302-313.

[HE&Al-2] M. Heinrich, J. Kuskin, D. Ofelt, J. Heinlein, J. Baxter, J. Pal Singh; R. Simoni, K. Gharachorloo, D. Nakahira, M. Horowitz, A. Gupta, M. Rosenblum; J. Hennessy
« The performance impact of flexibility in the Stanford FLASH multiprocessor »
Proceedings of the 6st International Conference on Architectural Support for Programming Languages and Operation Systems. Octobre 1994, 23 pages.

[HEPA] J.Hennessy, D. Patterson
« Architecture des ordinateurs : Une approche quantitative »
Edition française par Daniel Etiemble et Michel Israël
McGraw-Hill, 1992, pages 121-124.

[HSL] IEEE
« IEEE Draft Std 1355 Standard for Heterogeneous Interconnect (HIC) Low Cost Low Latency Scalable Serial Interconnect for Parallel System Construction »
IEEE Standards Department. August 1994.

[INMOS] INMOS
« IMS C104 packet routing switch »
Inmos component datasheet. June 1993, 51 pages.

Bibliographie.

[AGER] T Agerwala, J.L. Martin, J.H. Mirza, D.C. Sadler, D.M. Dias, M. Snir
« SP2 system Architecture »
IBM systems Journal, Vol 34, N° 2, 1995, pages 153-184.

[BOUARA] A. Bouaraoua
Thèse de Doctorat de l'Université Paris VI en cours.
MASI, IBP.

[BLUMRICH]
M.A. Blumrich, K. LI, R. Alpert, C. Dubnicki, E.W. Felten
« Virtual Memory Mapped Network Interface for the SCHRIMP Multicomputer »
Proceedings of the 21st Annual Symposium on Computer Architecture, April 1994, 12 pages.

[CHAIKEN94]
J. Kubiatowicz, D. Chaiken, A. Agarwal
« The Alewife CMMU : Addressing the Mutliprocessor Cimmunication Gap »
HOTCHIPS 1994.

[**CHAIKEN95]**
David Chaiken , Anant Agarwal, Ricardo Bianchini and the ALEWIFE Team
« The MIT Alewife Machine : Architecture and performance »
Proceedings of the 22th International Symposium on Computer Architecture, 1995, 12 pages.

[CLARK] D. Clark, V Jacobson, J. Romkey and H. Salwen
« An Analysis of TCP Processing Overhead »
IEEE Communications Magazine 11,2, June 1989, pages 23-29.

[CYPHER] R. Cypher, A.Ho, S. Konstantinidou, R. Messina
« Architectural Requirements for Parallel Scientific Applications with Explicit Communication »
Proceedings of the 20th International Symposium on Computer Architecture. 1993, pages 2-13.

[DALLY87] W. Dally, C Seitz
« Deadlock-Free Message Routing in Multiprocessor »
IEEE Transaction on Computers, May 1987, pages 547-553.

[DALLY90] W. Dally
« Virtual Channel Flow Control »
Proceedings of the 17th International Symposium on Computer Architecture, 1990, pages 60-68.

Remerciements.

Si la rédaction d'une thèse est un exercice essentiellement personnel, les travaux présentés ici sont le fruit du travail d'une équipe. C'est pour cela que je tiens à remercier tous ceux qui y ont participé, en particulier Vincent Reibaldi, chargé de la micro-architecture et de l'implémentation matérielle de RCube, dont les travaux sont indissociables des miens, mais aussi Jean-Lou Desbarbieux et Frank Wajsbürt, qui se chargent de celle de PCI-DDC. Abdelhafid Bouaraoua a mis à notre disposition MILE, le simulateur de réseau, et Jean Jacques Lecler, qui a fortement participé à l'élaboration du protocole « Direct Deposit Stateless Receiver Protocol » a aussi effectué de nombreuses simulations sous Simscript.

Merci à Belkacem Zerrouk, le père de RCube, qui a effectué les études préalables pour le routeur et qui est à l'origine du projet. Vincent Reibaldi et moi-même avons bénéficié de son encadrement efficace dans nos travaux sur le routeur.

Merci à Paul Feautrier et à Daniel Etiemble d'avoir accepté d'être rapporteur de ma thèse, ainsi qu'à Isabelle Demeure, Roland Marbot, Gay Mazare et Pascal Sainrat pour avoir participé à mon jury.

Alain Greiner a apporté son expertise pour ce projet. Il a convaincu de nombreux industriels de la validité de notre approche. Je le remercie tout particulièrement pour son encadrement efficace et sa grande disponibilité.

Roland Marbot a encouragé notre projet en mettant à notre disposition la technologie HSL IEEE 1355 de Bull.

Paul Feautrier et Pierre David nous ont permis de mieux cerner les problèmes logiciels. Leur participation, avec Alexandre Fenyö, au projet MPC est un gage de succès.

Merci à Franck Cappello et Pierre David, pour leurs remarques constructives pendant les relectures de ce manuscrit.

Merci aux Ministres de la Recherche et aux Commissaires de la CEE, qui m'ont financé pendant ces trois années, ainsi qu'à tous ceux qui m'ont aidé à joindre les deux bouts.

Daniel Mondot et Denis Roux, qui m'ont fait découvrir l'informatique et l'électronique, furent d'excellents professeurs.

Merci enfin à mes proches de m'avoir soutenu durant cette période.

nous n'avons pas pu intégrer dans le cadre de ce premier projet, comme les canaux virtuels par exemple, et proposer de meilleurs caractéristiques, comme une valence plus grande ou une mémorisation plus importante. Ils bénéficieront par ailleurs des enseignements tirés des expériences effectuées sur RCube et PCI-DDC.

La disponibilité prochaine de notre système d'interconnexion, et sa mise en œuvre dans le cadre de la machine MPC permettra aussi une collaboration avec des équipes universitaires et industrielles sur les mécanismes logiciels à mettre en œuvre pour concevoir des machines parallèles autours de notre système. En effet, la machine MPC, que nous avons évoquée au chapitre 5, requiert des développements logiciels qui ne sont pas traités dans le cadre de ces travaux. Ces développements permettront aussi de définir éventuellement de nouveaux mécanismes matériels, à intégrer dans un futur réseau d'interconnexion, pour soulager le processeur de certaines tâches.

Conclusion.

Notre objectif était de construire un réseau d'interconnexion à fortes performances et haut débit pour machines parallèles.

Pour cela, nous avons conçu et réalisé le routeur de paquets RCube, utilisant la liaison 1 Gbit / sec. HSL, dont la fabrication est en cours. Le temps de routage d'un paquet est court (150 ns), en particulier grâce à l'utilisation du routage Wormhole. Ce composant met en œuvre la technique de routage par intervalles, qui permet d'utiliser des topologies quelconques et de garder des tables de routage de faible taille, quel que soit le nombre de nœuds présents dans le réseau. RCube présente en plus un mécanisme adaptatif général, qui permet le routage concurrent de paquets ayant la même destination sur des liens différents et qui implémente une tolérance aux pannes.

RCube permet la construction de réseau d'interconnexion à faible latence et haut débit pour des topologies comprenant un grand nombre de nœud.

Un composant d'interface entre le bus PCI et la liaison HSL a été spécifié, et sa micro-architecture définie. Sa réalisation matérielle est en cours, en utilisant un premier modèle VHDL déjà disponible. De faible complexité, PCI-DDC implémente notre protocole « Direct Deposit Stateless Receiver Protocol », qui constitue le point clef de notre système.

PCI-DDC remplie une fonction simple, mais essentielle : il assure le déplacement de portions de mémoire entre des nœuds distants, en évitant éventuellement les recopies inutiles, et en limitant les opérations logicielles nécessaires à l'envoi et à la réception des données.

Les récentes annonces des grands constructeurs, comme INTEL par exemple, qui proposent des jeux de composants PCI de plus en plus performants, en terme de bande passante et de concurrence d'accès à la mémoire entre les composants PCI et le processeur, nous confortent dans l'approche que nous avons eue. L'utilisation du bus PCI nous permet de bénéficier de l'augmentation de la puissance des cellules monoprocesseur, qui nécessite des développements tellement importants qu'ils sortent souvent du cadre des réalisations envisageables par une équipe universitaire européenne.

Des simulations effectuées avec des modèles précis des composants PCI-DDC et RCube ont montré l'efficacité des mécanismes matériels de transfert de données. En particulier le débit de notre système est élevé, et sa latence faible.

Nous avons donc montré qu'il était possible de développer un réseau d'interconnexion à très fortes performances, tout en gardant des opérateurs matériels simples, donc de faible coût.

Les performances de notre réseau de communication, en simulation, sont identiques, voire meilleures que celle de dispositifs comparables. Mais c'est la simplicité des réalisations, en nombre de composants, et la faible complexité des composants, en nombre de transistors, qui constituent le principal avantage de notre système.

En effet, le compromis entre les performances et le coût que propose notre réseau est particulièrement intéressant. Une carte PCI contenant un RCube et un PCI-DDC est comparable, en terme de consommation, de connectique, de nombre de transistors, d'occupation en surface, de technologie, c'est-à-dire de coût, à une carte réseau PCI actuelle.

Ces deux composants, RCube et PCI-DDC, constituent une étape dans la démarche entreprise par notre équipe autour du standard IEEE 1355. Leur validation, ainsi que leur mise en oeuvre, permettront d'acquérir l'expérience nécessaire au développement éventuel d'un second routeur, et d'un second composant d'interface. Ceux-ci devront proposer les mécanismes que

simulations montrent en tout cas que la charge des liens peut dépasser 40 pour-cent pour 64 processeurs.

Pour les grandes topologies, il peut être intéressant d'augmenter plus rapidement le nombre de routeurs que celui de processeurs. Pour ne pas avoir à développer des cartes indépendantes, on peut placer deux cartes PCI-DDC / RCube dans certaines cartes mères. L'une d'entre elles sert d'interface au réseau pour le processeur, tandis que sur l'autre, le composant PCI-DDC ne sert qu'à configurer le routeur, dont les huit liens sont alors utilisables. Ceci permet, pour des topologies avec un grand nombre de processeurs, de rajouter des routeurs pour permettre l'augmentation du taux de charge où la saturation apparaît.

Le composant d'interface PCI-DDC combine les avantages des différents systèmes examinés dans le chapitre 2.

Il propose un mécanisme de reconstruction des paquets en messages, qui permet d'éviter un traitement logiciel à chaque paquet.

Les données ne sont pas mémorisées dans l'interface en émission, mais envoyées au fur et à mesure que le composant les prend depuis la mémoire centrale. Evidemment, cela se fait au prix de concurrences d'accès avec le processeur à la mémoire, mais peut-il en être autrement ? Le composant assure seul la décomposition des fractions contiguës en mémoire du message en paquets.

Système	Interface Applicative	Intégration	% bande passante utilisable (bidirectionnel)	Latence Matérielle (copie 32 octets)
CRAY T3D	Sans appels système, mais sur Machine dédiée	4 composants	48 % de 150 Mo[46]	2,7 µs
Myrinet	Appels système	4 composants	38 % de 80 Mo	20 µs
IBM SP2	Sans appels système	7 composants	52 % de 40 Mo	40 µs (1 mot)
PCI-DDC	Appels système	2 composant, routeur inclus	50 % de 80 Mo	3 µs

Figure 6-o : comparaison PCI-DDC / état de l'art (2).

Les performances simulées de notre système sont supérieures à celles de Myrinet. Si les mesures expérimentales confirment les simulations, nous aurons atteint notre objectif : construire un réseau pour machines parallèles à base de stations de travail à très fortes performances et très bas coût.

6-5 Conclusion.

Nous avons vu que le composant PCI-DDC est capable de générer une charge HSL jusque 50 pour-cent. Cette charge est légèrement supérieure à ce que le réseau composé de RCube est capable de router, car celui-ci, dès 40 pour-cent de charge, est saturé. Les recherches en cours doivent déterminer d'autres topologies pour lesquelles le taux de charge applicable sans congestion est plus élevé.

Le point d'équilibre dans un réseau d'interconnexion peut être simplement trouvé en ajustant la taille maximale des paquets. Plus ceux-ci sont petits, plus le débit généré par le composant PCI-DDC baisse, et plus le taux de charge de saturation augmente. Plus les paquets sont gros, plus le débit augmente et le taux de charge de saturation diminue.

L'utilisation du bus PCI par d'autres ressources que notre composant PCI-DDC limite aussi la taille des paquets, par le mécanisme de « Latency Timer ». C'est pour cela que ces réglages ne sont possibles que sur un système fonctionnant, grâce à des expériences réelles. Nos

[46] 73 Mo / sec par Alpha. Comme deux Alpha partage un même lien, on compare cette valeur à 150 Mo / sec de débit crête et non 300 Mo / sec.

La technique de routage par intervalles permet une nette réduction de la taille des en-têtes des paquets par rapport au routage explicite. L'utilisation de liens série est plus souple que celle de liens parallèle et permet des routages sur des distances plus importantes. Le support matériel de ces liens série (micro câble coaxial) est par ailleurs peu coûteux.

Système	Commutation	Interblocage (1)	Adaptabilité (2)	Contention (3)
CRAY T3D	Incomplète	Doublement des « mémoires tampons »	Non	Non
Myrinet	Crossbar complet	Non	Non	Non
INMOS C104	Crossbar complet	Non	« Grouping »	Non
IBM SP2	Crossbar complet	Non	Non	Buffered Wormhole
RCube	Crossbar complet	Non	Générale	Non

Figure 6-m : comparaison RCube / état de l'art (2).

(1) : Présence d'un support matériel spécifique pour éviter les interblocages.
(2) : Présence d'un support matériel spécifique pour permettre l'adaptabilité.
(3) : Présence d'un support matériel spécifique pour permettre le règlement des contentions.

Myrinet et le C104, par l'utilisation du routage « wormhole », présentent certainement des charges maximales applicables avant la saturation du réseau, similaires à celle de RCube. Celle du T3D est inférieure, car la technique de routage, là aussi, est « wormhole », et les contentions sur le routage concurrent dans l'élément de commutation incomplet doivent bloquer des paquets qui auraient pu être routés dans un routeur avec un « crossbar » complet. Le routeur du SP2 doit par contre, grâce à l'utilisation de sa mémoire interne dans le cadre du « Buffered Wormhole Routing », permettre une meilleure tenue aux charges plus élevées du système, augmentant ainsi son seuil de saturation.

6-4-1 Interface et protocole.

Système	Interruption en réception	Acquittement et Réémission	Mémorisation en émission	Copie logicielle	Traitement logiciel en réception
CRAY T3D	Par paquet	Oui	Non	1	Extraction par le processeur des données.
Myrinet	Par paquet	Non	Oui	0	Aucun
IBM SP2	Par message	Oui	Non	0	Aucun
PCI-DDC	Par message	Non	Non	0	Aucun

Figure 6-n : comparaison PCI-DDC / état de l'art (1).

Dans les deux tableaux 6-n et 6-o, nous présentons une comparaison entre PCI-DDC et les différentes interfaces présentées dans le chapitre 2.

On va donc utiliser une file de 192 octets, ce qui limite environ la taille des paquets à 1000 octets, le surcoût de protocole est alors de moins de 3 pour-cent, c'est-à-dire du même ordre que le contrôle de flux.

File d'attente de réception.

Il est difficile de dimensionner cette file d'attente avec les simulations que nous avons faites. En effet, dans le cas de deux composants PCI-DDC dont le trafic est équilibré, cette file d'attente se remplit statistiquement au même débit qu'elle se vide, car la charge s'équilibre.

Si on envisage un cas où un composant PCI-DDC reçoit simultanément plusieurs messages provenant de différents interlocuteurs, au débit maximum de 80 MOctets / sec., quelle que soit sa taille, elle sera rapidement saturée.

L'intérêt principal de cette file d'attente est de pouvoir, dans le cas d'un « pic » de réception de paquets, de pouvoir extraire les données du réseau pour éviter de l'engorger. Evidemment, si ce « pic » se prolonge, le réseau s'engorgera.

Nous allons insérer une file de plus grande taille possible, tout en restant raisonnable d'un point de vue de la surface de silicium. Une taille de file de 576 octets (512 octets dans la file synchrone, 64 dans la file asynchrone) est pour le moment la solution retenue, compte tenu des contraintes de coût de réalisation.

6-4 Comparaison.

Dans cette dernière section, nous effectuons une comparaison rapide de notre système d'interconnexion avec les différents systèmes d'interconnexion présentés dans le chapitre 2.

6-4-1 Eléments de routage.

Sur les deux tableaux ci-dessous, nous résumons les caractéristiques des éléments de routage que nous avons présentés dans le chapitre 2, ainsi que RCube.

Système	Liens	Débit crête	Latence	Routage	Topologie
CRAY T3D	6 liens parallèle	300 Mo	14 ns	Routage explicite	Tore 3D
Myrinet	8 liens parallèle	80 Mo	550 ns	Routage explicite	Quelconque
INMOS C104	32 liens série	10 Mo	1000 ns	Routage par intervalles	Quelconque
IBM SP2	8 liens parallèle	40 Mo	125 ns	Routage explicite	Quelconque
RCube	8 liens série	80 Mo	150 ns	Routage par intervalles	Quelconque

Figure 6-l : comparaison RCube / état de l'art (1).

Les simulations réalisées montrent que le potentiel de performances de RCube est du même ordre, sinon meilleur, que les performances mesurées des réalisations existantes. Seul le routeur du CRAY T3D possède un débit supérieur, dû en partie à l'utilisation d'une distribution unique d'horloge, qui est extrêmement coûteuse. Le routeur du SP2 présente une latence plus faible que celle de RCube.

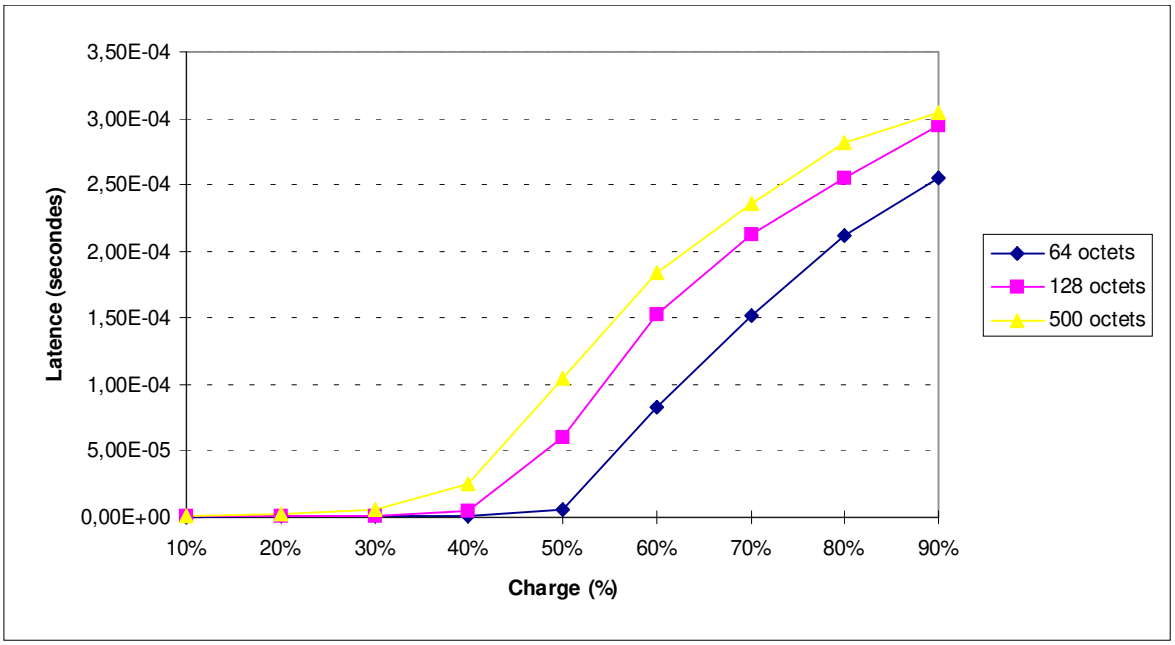

Figure 6-k : Latence de routage et taille de paquet.

6-3-3 Profondeur des files d'attente.

La question du dimensionnement des files d'attente d'émission et de réception se pose aussi. Il s'agit de trouver un compromis entre le coût (c'est-à-dire la surface de silicium) et le gain de performances qu'elles apportent.

File d'attente d'émission.

On peut distinguer deux cas dans le fonctionnement de l'émetteur de PCI-DDC.

Dans le cas où il n'y a pas de contention sur le lien HSL, et où une émission est en cours, la file d'émission se remplit à environ 100 MOctets / sec. et se vide à 80 MOctets / sec. Lorsqu'elle se remplit entièrement, comme nous l'avons vu dans les chapitres 3 et 4, le paquet est clos. Ce mécanisme introduit donc une limitation implicite de la taille des paquets, qui n'est pas gênante en soi, comme nous l'avons vu dans la section précédente.

Sa taille minimale est de 64 octets de file asynchrone (deux « flits »), pour permettre la gestion du contrôle de flux, plus 16 octets de file synchrone, pour permettre l'écriture de la fin du paquet lorsque la file est pleine, soit un total de 80 octets. Cette file virtuelle de 80 octets fixe donc, avec les débits définis ci-dessus, une taille maximale de paquets de 400 octets. Cette taille est un peu trop faible, car nous avons 25 octets de surcoût de protocole par paquet, ce qui représente ici 6 pour-cent.

Un second cas est celui où la file d'attente se remplit entièrement à cause d'une contention sur le réseau qui empêche la tête du paquet d'être routée. Dans ce cas, il faut interrompre l'émission le plus rapidement possible, et privilégier la réception de paquets. La fin du paquet en émission est dans la file d'attente, et la bande passante est utilisée à émettre des caractères « IDLE » inutiles. C'est le phénomène que nous avons vu apparaître sur les figures 6-i et 6-j. Bref, dans ce cas, plus la file est petite, plus la fin du paquet est envoyée rapidement.

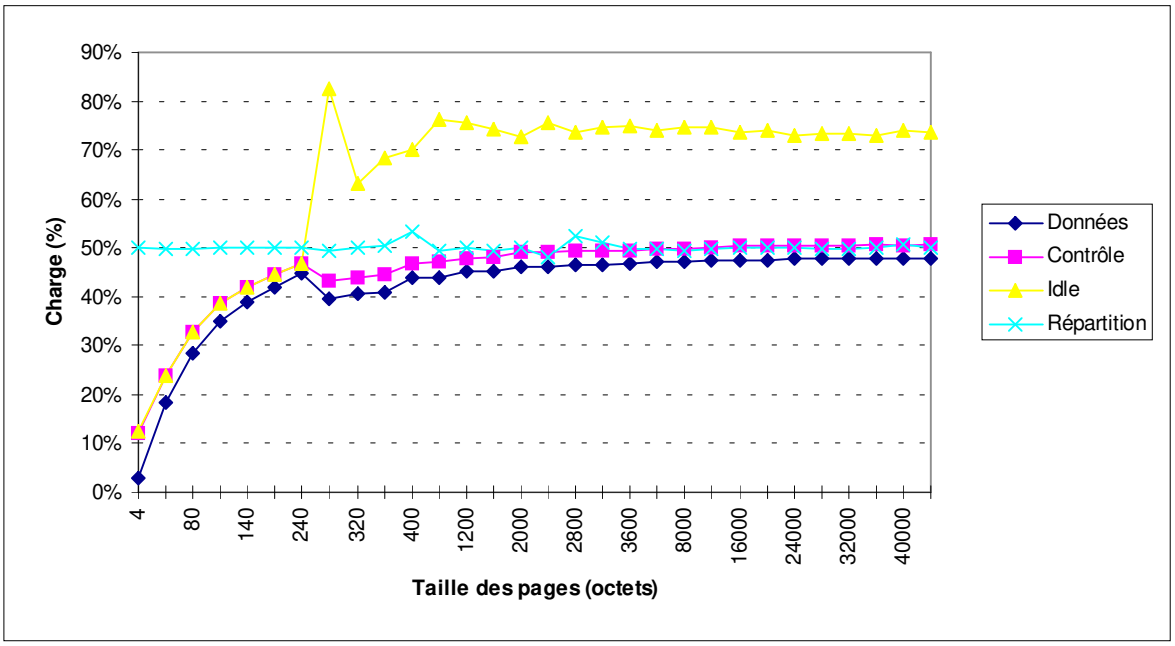

figure 6-j : simulation avec PCI-DDC en priorité tournante.

Dans le cas de notre simulation de la figure 6-b, où les demandes de trafic des deux composants sont égales, la valeur de *N* est de 4, ce qui constitue la valeur par défaut de PCI-DDC. On a pu voir que les deux phénomènes que nous venons d'exposer ne s'y présentaient pas : peu de caractères « IDLE » sont émis, et le trafic est équilibré.

La pratique et l'expérience sur un système en fonctionnement sur des applications réelles fourniront des résultats plus précis sur l'influence de ce paramètre, dont le positionnement pourra être ensuite intégré au sein du composant.

6-3-2 Limitation de la taille des paquets.

Une autre expérience sur le réseau de routage nous a permis d'établir une relation entre la latence de routage et la taille des paquets présents sur le réseau. La figure 6-k présente des simulations de notre grille composée à deux dimensions de 64 processeurs. On y mesure la latence en fonction de la charge, pour différentes tailles de paquets.

On remarque que la latence moyenne croît lorsque la taille des paquets augmente. Ceci est dû au fait que plus les paquets sont « longs », plus leur blocage dans un routeur pénalise les autres communications. La taille des paquets influe de manière forte sur le seuil de saturation. Sur la figure 6-i, le taux de charge pour arriver à une latence moyenne de 500 cycles varie de 30% à 50% suivant la taille maximale des paquets. On a donc intérêt à limiter leur taille à la génération, c'est-à-dire dans le composant PCI-DDC.

PCI-DDC propose un registre dans lequel on peut fixer une borne maximale à la taille des paquets. Celui-ci peut aussi être modifié dynamiquement, pour permettre sa reconfiguration par le système d'exploitation en cours de fonctionnement.

Il faut bien évidemment trouver un compromis entre la perte due à la répétition des en-têtes et des codes détecteurs d'erreurs dans chaque paquet d'une page, et la perte due aux paquets trop longs. La valeur par défaut est de 512 octets.

Là aussi, l'expérience sur le système complet permettra de fixer une valeur optimale, en fonction des topologies et du type de trafic généré. Le positionnement de cette valeur, lui aussi, pourra être intégré au composant par la suite.

La figure 6-i présente les résultats d'une simulation avec réception prioritaire.

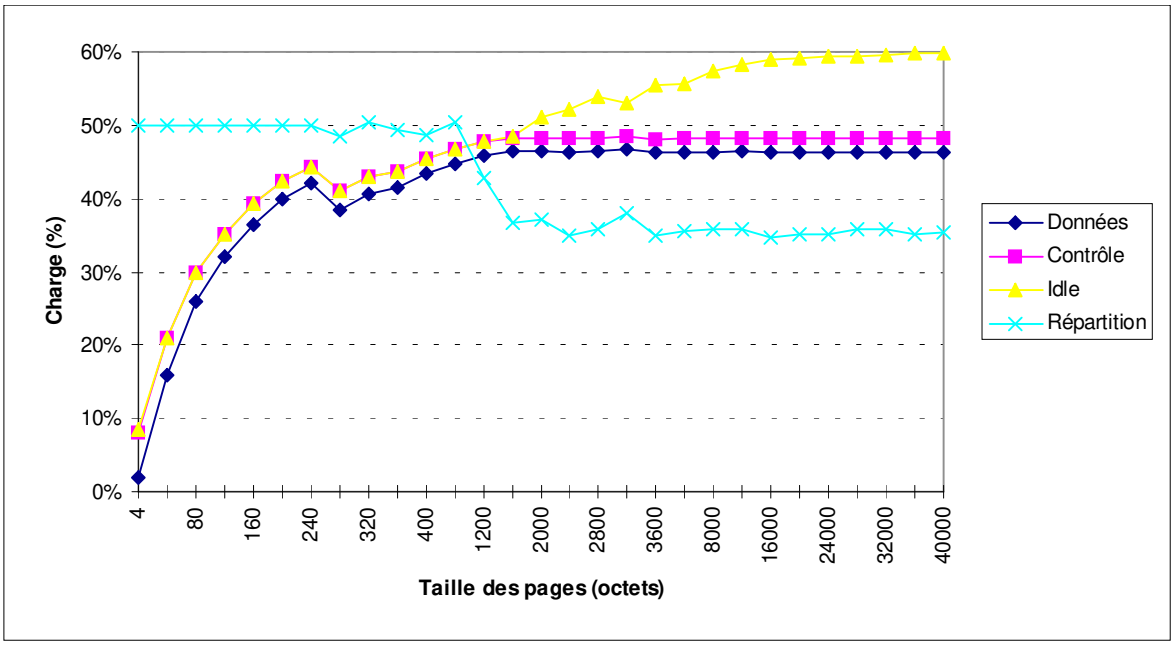

Figure 6-i : simulation avec PCI-DDC en réception prioritaire.

Ce qui était prévisible apparaît : nos simulations montrent que le trafic ne se répartit pas équitablement entre les deux processeurs (suivre la courbe « Répartition »). L'un des processeurs émet environ deux fois plus de données que l'autre (35% contre 65%).

De plus, le taux de transfert de données (courbe "données"), qui totalise les données émises dans les deux sens, est légèrement plus faible que sur la figure 6-b.

Enfin, et c'est là le plus grave, comme le trafic est fortement asymétrique, la file d'attente de réception de l'un des composants est saturée, d'où l'apparition de nombreux caractères « IDLE », insérés dans un message en cours, qui rajoutent inutilement de la charge sur le lien.

Sur la figure 6-j, on observe la courbe de notre deuxième choix, avec une priorité tournante entre l'émission et la réception.

Ici, la répartition entre les deux intervenants est équitable, mais un grand nombre de caractères « IDLE » sont émis, car les files d'attente de réception débordent.

Bref, il faut effectuer un compromis entre ces deux situations extrêmes.

Il faut éviter les famines, qui génèrent des situations asymétriques, il faut cependant privilégier la réception, pour libérer le plus rapidement possible le réseau d'interconnexion des paquets qui y transitent.

On utilise donc une priorité légèrement plus complexe que les deux cas extrêmes présentés ci-dessus, avec un paramètre *N* : après une réception de paquet, *N* fois sur 16, PCI-DDC cherche d'abord à effectuer une autre réception et *(16-N)* fois sur 16 il va émettre un paquet si des émissions sont programmées.

Le paramètre N, fixé par logiciel, peut être modifié dynamiquement au cours du fonctionnement du composant. Le système d'exploitation, suivant l'utilisation qui est faite du composant PCI-DDC, c'est-à-dire la répartition entre les émissions et les réceptions, peut changer ce paramètre pour obtenir de meilleures performances.

Charge	Adaptatif	Déterministe.
0,1	616 cycles	644 cycles
0,2	779 cycles	882 cycles
0,3	1764 cycles	1918 cycles

Figure 6-h : Evaluation du gain de l'adaptabilité sur une grille.

Ce résultat s'explique par le fait que nos simulations supposent que le débit est spatialement réparti. Il est donc clair que l'adaptabilité, c'est-à-dire le report de charge d'un chemin vers d'autres chemins n'est pas très pertinent dans ce cadre, car tous les chemins sont autant chargés les uns que les autres.

Qualitatifs :

Le gain de l'adaptabilité doit être mesuré dans des cas où les communications ne sont pas réparties de manière égale entre les nœuds, mais où certains dialoguent plus que d'autres. Dans ce cas, les liens plus chargés que d'autres pourront reporter une partie de leur charge sur les liens voisins. On verra alors, application par application, le gain effectivement généré par la mise en place de tables de routage adaptable dans le réseau, et juger finalement de l'efficacité du mécanisme dans certains cas précis.

Finalement, la justification de la mise en place du mécanisme d'adaptabilité tient dans le fait que nous avons choisi ce mécanisme pour implémenter la tolérance aux pannes sur un lien. Le routeur choisissant, parmi les liens possibles pour un intervalle, celui de plus forte priorité qui est fonctionnel, l'adaptabilité permet de configurer les sorties alternatives à choisir, en cas de panne du lien principal.

6-3 Optimisations.

Les performances présentées ci-dessus incluent quelques paramètres « fins » dont nous présentons la détermination maintenant.

6-3-1 Priorités entre l'émission et la réception.

Puisque le bus PCI ne permet les transferts qu'alternativement dans un sens et dans l'autre (soit en émission, soit en réception), il faut choisir une politique de priorité entre la réception et l'émission. La figure 6-a présente la modélisation de notre automate. On peut voir deux choix différents :

- La réception est prioritaire : après la réception d'un paquet, on vérifie qu'un autre n'est pas présent dans la file d'attente de réception avant de tenter d'émettre.
- La priorité est tournante : c'est-à-dire qu'après chaque réception de paquet, on regarde s'il n'y a pas un paquet à émettre et, le cas échéant, on effectue son émission.

Le premier choix présente un avantage et un inconvénient.

Le réseau RCube et sa technique de routage « wormhole » nous incitent à retirer les paquets le plus rapidement possible du réseau, pour permettre des routages concurrents dans les routeurs traversés. On est tenté de privilégier la réception pour éviter les congestions.

Le problème est qu'il existe alors un risque de famine : un nœud qui reçoit des paquets en permanence n'est pas en mesure d'en émettre.

processeurs actuels. Un P6 à 160 Mhz exécute de 300 à 400 instructions en 1,2 microsecondes.

- Le temps de routage d'un octet est de l'ordre de 400 cycles, soit 4,8 microsecondes dans un réseau à 64 processeurs.

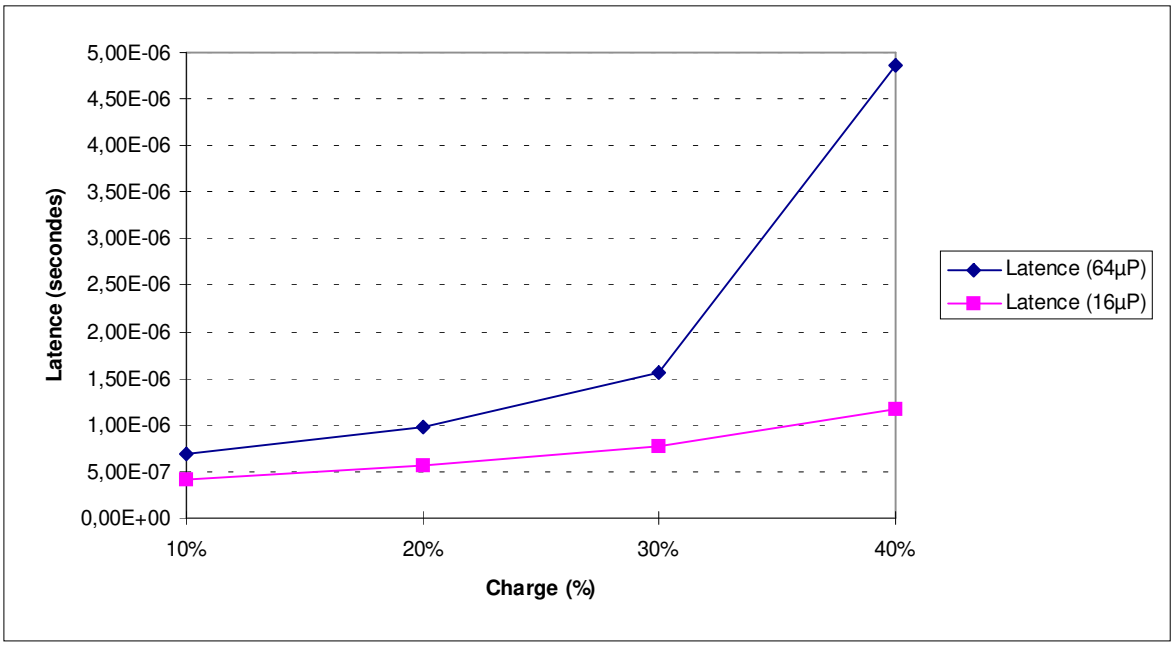

Figure 6-g : Détail de la courbe précédente.

L'étude de la topologie du réseau de notre machine est en cours. La recherche porte sur un réseau pour lequel « l'explosion » de la latence apparaît pour des charges plus élevées. Dans cette direction, la topologie utilisée dans le SP2 présente de bons résultats, même pour des réseaux à 128 processeurs. On peut se référer à notre article « Un routeur de messages adaptatif »[45] pour une évaluation de RCube sur cette topologie.

6-2-5 Evaluation du gain dû à l'adaptabilité.

Les gains dûs à la mise en œuvre de l'adaptabilité dans le routeur RCube sont de deux ordres : qualitatifs et quantitatifs.

Quantitatifs :

Les gains quantitatifs, mesurés sur nos simulations, sont forts modestes. Le tableau ci-dessous présente les résultats obtenus, en nombre de cycles par rapport à la charge (en pour-cent) sur une grille. Dans le meilleur des cas, la mise en œuvre de tables de routage adaptatives, par rapport à la mise en œuvre de tables déterministes, permet de réaliser un gain de l'ordre de 10 pour-cent sur le délai nécessaire au routage. Ce résultat n'est pas vraiment satisfaisant, car l'adaptabilité fait perdre un cycle de latence dans chaque RCube, et a augmenté la complexité du routeur. La mise en place d'une technique non-adaptable aurait fait passer le temps de routage de 150 ns à 125 ns.

[45] [ZERR96]

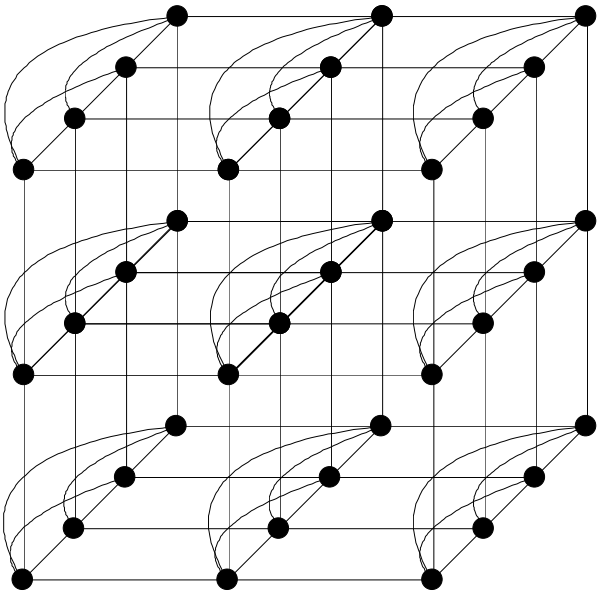

Figure 6-e : grille composée à deux dimensions.

On voit sur cette courbe, ce qui était prévisible, que le réseau supporte très mal les taux de charges élevés. En effet, le routage wormhole répercute sur tout le réseau les contentions qui peuvent apparaître sur un nœud, dès que la charge dépasse 40 ou 50 pour-cent. La mise en place d'un mécanisme utilisant les canaux virtuels aurait fortement amélioré cette situation. La figure 6-g présente un détail de cette courbe.

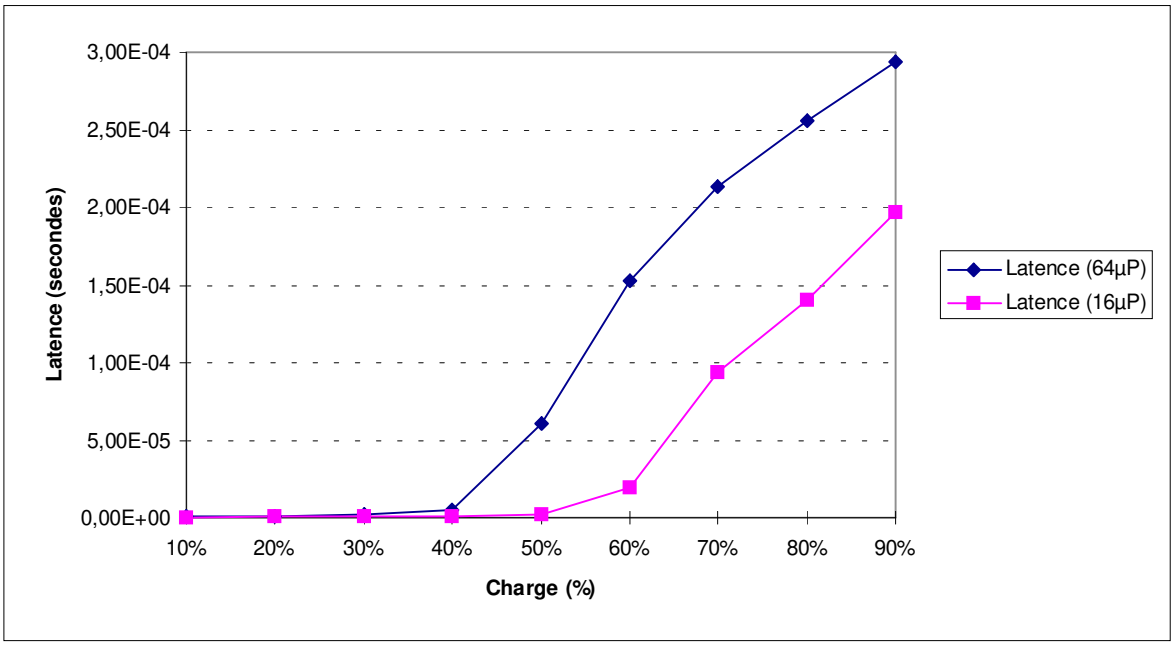

Figure 6-f : latence du transfert d'un paquet pour deux tailles de réseau.

Pour des charges inférieures à 40 pour-cent, les deux réseaux se comportent très bien.

- Le temps de routage d'un octet est de l'ordre de 100 cycles, soit 1,2 microseconde dans un réseau à 16 processeurs. Ce temps est encore élevé comparé aux performances des

6-2-3 Présentation de la simulation d'un réseau de RCube.

La simulation du réseau d'interconnexion s'effectue dans un autre environnement, MILE. MILE a été mis au point par Abdelhafid Bouaraoua et Belkacem Zerrouk et permet la simulation de réseaux d'interconnexion.

Nous avons écrit un modèle de RCube, en langage C, qui simule son fonctionnement au cycle près. Ce modèle s'intègre dans l'environnement de simulation MILE pour permettre la simulation d'un réseau. Dans notre simulation, un modèle de générateur de trafic envoie des messages de taille bornée, suivant une certaine distribution.

Ce générateur reflète le comportement d'un composant PCI-DDC. Comme on le voit sur la figure 6-d, deux paramètres sont définis :

- La taille maximale des paquets.
- L'occupation de la bande passante, ou la charge, qui est le pourcentage de caractères émis par rapport à la bande passante maximale.

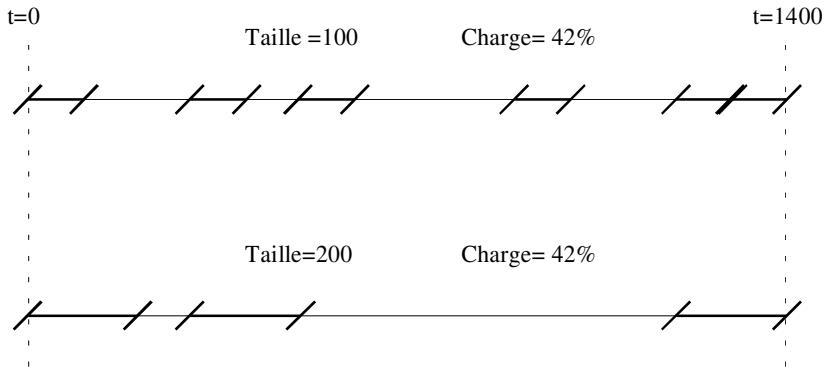

Figure 6-d : génération de paquets dans la simulation MILE.

Cette modélisation correspond effectivement à la réalité, car nous allons voir dans la section 6-3-2 qu'il faut limiter la taille des paquets.

La topologie que nous avons choisi pour mesurer les performances de notre réseau est une grille à deux dimensions composée. On peut voir sur la figure 6-e un exemple de cette grille pour 36 processeurs. Les tables de routage par intervalles et le marquage des nœuds sont basés sur le principe du « Turn Model » et proposent un routage adaptatif. Les simulations ont été effectuées par Abdelhafid Bouaraoua. Une étude beaucoup plus complète, portant sur différentes topologies et différentes stratégies de routage, sera présentée dans la thèse de M. Bouaraoua.

La « grille composée à deux dimensions » est composée de quatre grilles, les 4 points de mêmes coordonnées (x, y) sont complètement interconnectés. La valence maximale d'un nœud est de 7, ce qui est le cas des nœuds MPC.

6-2-4 Débit et latence d'un réseau RCube.

Sur la figure 6-f, on mesure la latence de cheminement d'un paquet de 128 octets à travers deux réseaux du type décrit ci-dessus, comportant respectivement 16 et 64 processeurs. Cette fois-ci, la latence est mesurée entre l'écriture du premier octet dans la file d'attente de l'émetteur et son arrivée.

pages inférieures, un seul paquet est émis, et pour des tailles supérieures, un deuxième paquet est émis, d'où une perte de bande passante. Pour les tailles encore supérieures, de nombreux paquets vont être émis, mais les irrégularités ne sont plus visibles sur la courbe.

La taille « limite » de 64 mots reflète la valeur du « Latency Timer » du composant PCI, qui est paramétrable entre 0 et 256. On peut se référer à la spécification 2.1 du bus PCI pour plus de détails sur ce paramètre. On dira juste que si une application doit transférer des tailles de pages d'environ 256 octets, il serait judicieux d'augmenter légèrement la valeur du « Latency Timer » (fixée par défaut à 64).

La figure 6-c donne des mesures de « latence » lors de l'émission de pages. La latence a été mesurée entre le début du transfert d'une page (prise du descripteur dans la LPE) et la fin du transfert d'une page (génération de l'interruption sur la machine en réception). Chacun des points du graphique donne, en ordonnée, une mesure de latence pendant une simulation pour une page de taille indiquée en abscisse. Les échelles verticale et horizontale sont logarithmiques.

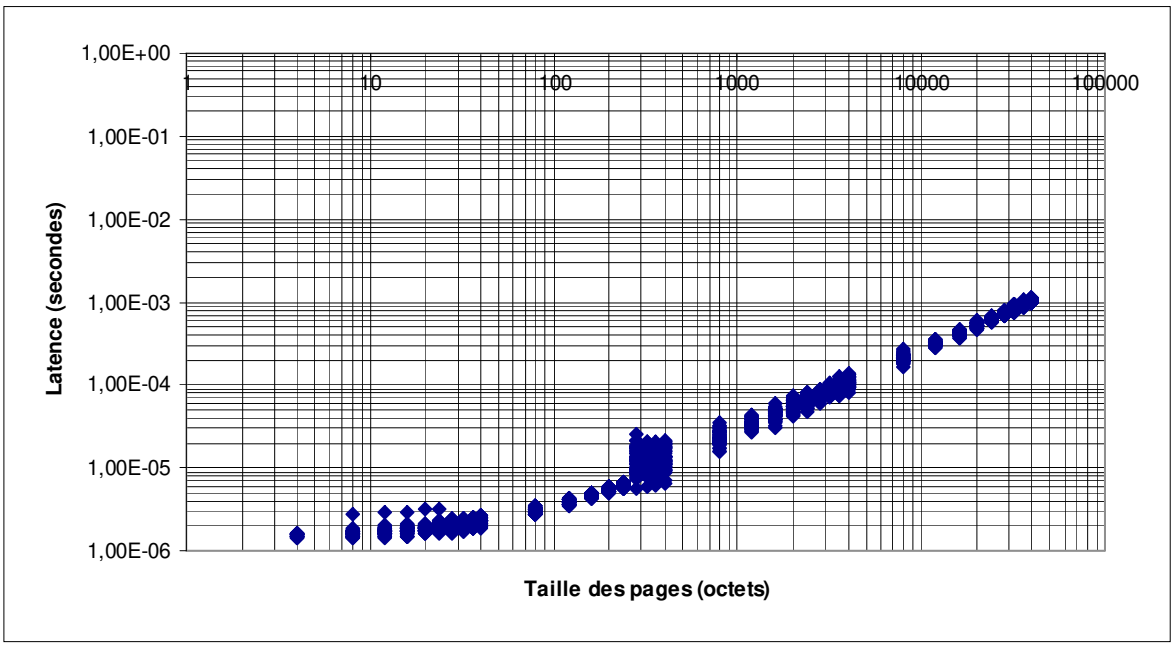

Figure 6-c : latence d'un transfert en fonction de la taille des pages.

On peut lire les informations suivantes :

- La latence matérielle du transfert d'une page de 4 KOctets est d'environ 100 microsecondes. La latence pour le transfert d'un message de signalisation de 20 octets est d'environ 2 microsecondes.
- La latence est une fonction affine de la taille du message dès que celui-ci a une taille supérieure à environ 400 octets.

Le phénomène irrégulier pour les tailles de pages de 256 octets est, là aussi, dû au « Latency Timer ».

l'émission et la réception, alors que celui d'un lien HSL bidirectionnel est de 160 MOctets / sec.

Sur la figure 6-b, nous représentons le pourcentage de bande passante HSL occupée par les paquets provenant des composants PCI-DDC en fonction de la taille des pages. La taille est donnée en nombre d'octets. Les pages désignent ici des zones adjacentes de mémoire, et non des pages de mémoire virtuelle.

La première courbe « Données » représente le nombre de caractères de données effectivement émis. La seconde courbe « Contrôle » totalise les caractères de données avec les caractères supplémentaires du paquet, c'est-à-dire les en-têtes et terminateurs que nous lui rajoutons; la troisième courbe représente les caractères « IDLE » insérés par le composant PCI-DDC au cours d'un transfert, au sein d'un paquet. La quatrième courbe indique l'équilibre entre les transferts effectués par les deux composants PCI-DDC. La valeur 50% signifie que les deux composants transfèrent des données dans les mêmes proportions.

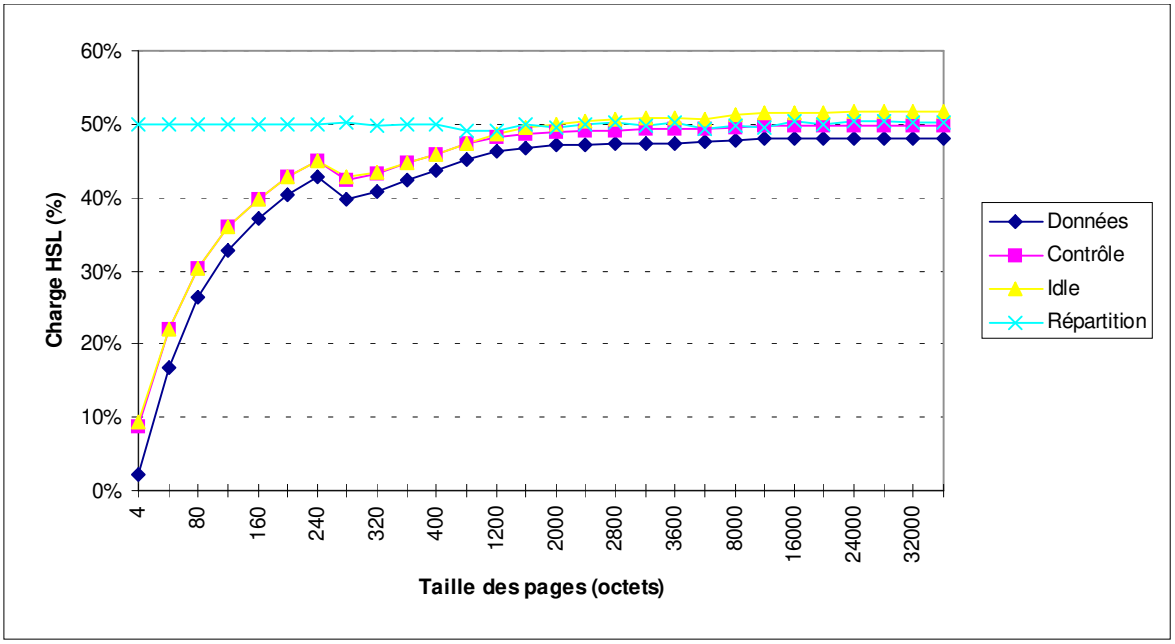

Figure 6-b : charge des liens en fonction de la taille des pages.

On peut effectuer les remarques suivantes sur ces courbes :

- Le débit maximum est 40 MOctets / sec. (50 pour-cent de 80 Mo / sec.) dans chaque direction. C'est-à-dire que les composants sont capables d'envoyer et de recevoir 40 MOctets de messages par seconde. La limitation est due ici au bus PCI.
- Le débit maximum est atteint lorsque les pages ont une taille supérieure à 2 KOctets. Pour des tailles de pages de 256 octets, le débit est d'environ 32 MOctets / sec. Le système est aussi capable de transférer 1 million de messages de signalisation de 20 octets par seconde.
- Le trafic est équitablement réparti entre les deux composants PCI-DDC (voir paragraphe 6-3-1).
- On obtient la moitié de la bande passante maximale pour des tailles de messages de 80 octets.

Sur cette courbe, on peut voir un phénomène irrégulier pour des messages d'une taille d'environ 256 octets. Celui-ci est dû à l'apparition d'un deuxième paquet. Pour les tailles de

celles relevées expérimentalement par certains utilisateurs, et rapportées dans les forums de discussion Internet PCI)

Le bus peut être utilisé aussi par d'autres périphériques. Ceci réduit le débit utilisable pour les communications entre les nœuds du réseau. Nous avons effectué des simulations avec une charge générée par d'autres composants PCI, mais nous ne les présenterons pas ici.

Modélisation de PCI-DDC.

Le composant PCI-DDC est modélisé en trois entités distinctes : un automate principal et deux files d'attente, l'une étant dédiée, comme dans l'implémentation physique, à la réception, l'autre à l'émission. L'automate est représenté sur la figure 6-a.

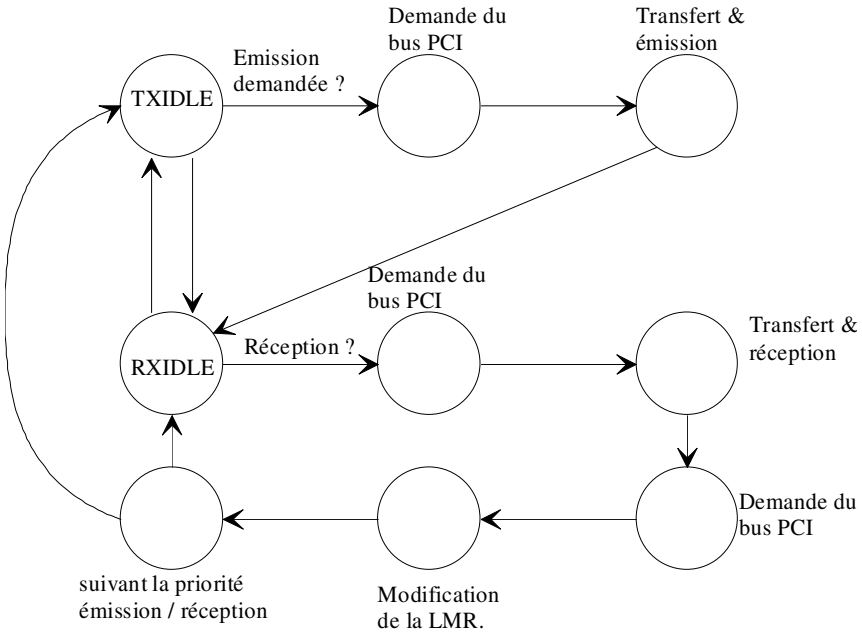

Figure 6-a : automate de PCI-DDC.

Au repos, il oscille entre deux états desquels il peut partir pour effectuer des opérations d'émission ou de réception.

Dans l'état « TXIDLE », si une entrée est présente dans la LPE, ou si un paquet d'un message en cours doit être envoyé, l'automate effectue un cycle d'émission de paquets.

Dans l'état « RXIDLE », si des données sont présentes dans la file d'attente de réception, l'automate effectue la réception du paquet, puis la mise à jour de la Liste des Messages en Réception. Ensuite, suivant une politique que nous décrivons plus loin, il va privilégier, dans le cas d'émissions et de réceptions concurrentes, soit l'émission, soit la réception.

Le composant PCI-DDC est simulé en tenant fidèlement compte du nombre de cycles de chaque opération, tel qu'il est décrit dans le premier modèle VHDL que nous avons réalisé.

6-2-2 Débit et latence de PCI-DDC.

Nous allons, dans un premier temps, mesurer la bande passante du lien HSL occupée par les paquets générés par un composant PCI-DDC. Il est en effet clair que le composant PCI-DDC ne peut saturer les liens HSL, car le débit du bus PCI est de 132 MOctets / sec. en crête pour

PCI-DDC sont capables de générer, sans être ralentis par le réseau, et ensuite nous vérifions que le réseau est capable de router ce trafic, sans être congestionné.

Cependant, cette simulation en deux temps n'a évidemment pas la même force de preuve qu'une simulation globale, où les composants, tels qu'ils sont réalisés, auraient été simulés ensemble. Cette simulation globale était difficilement réalisable, dans la mesure où les temps de calculs auraient été beaucoup trop importants. Il faut déjà une dizaine d'heures pour simuler 1 seconde de fonctionnement de deux composants PCI-DDC. Nous n'avions pas les moyens matériels d'en simuler 64 en même temps, alors que nous avons pu simuler 64 processeurs, interconnectés par un réseau RCube, et ayant un comportement statistiquement identique à celui obtenu par la simulation de deux PCI-DDC reliés entre eux sans routeur.

Dans la section 6-2, nous présentons les deux simulations et leurs résultats, après le réglage des paramètres. Dans la section 6-3 nous présentons les expériences qui nous permis de régler ces paramètres et enfin dans la section 6-4 nous effectuons des comparaisons avec les systèmes présentés dans le chapitre 2.

6-2 Résultats.

6-2-1 Présentation de la simulation PCI-DDC.

Simscript.

La première simulation a été effectuée en utilisant le logiciel Simscript. Celui-ci est un environnement de simulation à temps continu. La description de PCI-DDC a été effectuée à l'aide d'un langage de haut niveau, en utilisant des objets définis dans Simscript, comme les files d'attente par exemple. Ce langage procédural, qui ressemble à du VHDL haut niveau, permet de voir évoluer simultanément des automates, et d'effectuer des mesures sur différents événements. On a donc simulé un modèle comprenant un bus PCI et un composant PCI-DDC pour permettre la mesure des performances attendues.

Bus PCI.

Il s'agit de simuler le comportement d'un composant PCI-DDC sur un bus PCI. La modélisation d'un bus PCI n'est pas une mince affaire. Elle dépend considérablement du jeu de composants qui constituent le bus PCI, et les constructeurs ne fournissent pas un modèle précis des composants qu'ils proposent.

Nous avons donc effectué les hypothèses suivantes :

- Le bus PCI est cadencé à 33 Mhz sur 32 bits.
- Lorsqu'un composant demande le bus pour y effectuer des accès en mode maître, et que celui-ci n'est pas occupé par un autre composant, l'affectation du bus PCI au composant qui le demande par l'arbitre est effectuée avec le nombre minimal de cycles prévu par la spécification PCI.
- Lorsqu'un accès est effectué en mémoire principale par un composant PCI, la mémoire principale introduit des états d'attente. On considère qu'un cycle sur 4, lors d'une rafale, est un état d'attente, ce qui positionne le débit du bus, lors d'un accès rafale, à 100 MOctets / sec., au lieu des 132 MOctets / sec. théoriques. (Cette valeur correspond aussi à

Chapitre 6 Analyse des performances.

Nous avons présenté dans les chapitres 3 et 4 les principes de base de notre architecture, les deux composants RCube et PCI-DDC, ainsi que le mécanisme fondamental de communication, le « Direct Deposit Stateless Receiver Protocol ». Nous présentons ici différentes simulations permettant d'évaluer les performances attendues.

6-1 Conditions expérimentales.

6-1-1 Objectifs.

Nous allons chiffrer, par simulation, les performances des mécanismes **matériels** du réseau d'interconnexion en faisant les hypothèses suivantes :

- *Hypothèse 1* : Les applications qui s'exécutent sur chacun des nœuds demandent plus de communications que le matériel n'est capable d'en supporter, c'est-à-dire plus d'émissions et de réceptions de messages que le réseau d'interconnexion n'est capable d'acheminer. On considère que nos composants sont donc sollicités au maximum de leurs possibilités. On voit dans le paragraphe 6-2-1 comment ceci est réalisé. C'est un « pire cas » pour notre réseau d'interconnexion.
- *Hypothèse 2* : Les applications effectuent des échanges dont la distribution est équilibrée dans l'espace. C'est-à-dire que chacun des nœuds communique statistiquement de manière égale avec chacun des autres nœuds. Dans nos simulations, on considère donc que le système est totalement symétrique. La prise en compte de conditions asymétriques de trafic nous aurait écarté de notre objectif qui est la mesure des performances du matériel.

Ces simulations ont trois fonctions principales. Elles doivent d'un coté nous permettre de chiffrer les performances du dispositif, en particulier la latence et le débit. On doit aussi vérifier que les processeurs se partagent ce débit équitablement entre eux. Enfin, elles doivent nous permettre d'ajuster un certain nombre de paramètres « fins » de notre système, en particulier la taille des files d'attente et les priorités respectives de l'émission et de la réception de messages.

6-1-2 Mode opératoire.

Les simulations ont été effectuées en deux temps, de manière à séparer les problèmes. Dans une première simulation, nous avons relié deux composants PCI-DDC entre eux, sans aucun routeur, afin d'analyser le comportement des interfaces PCI. Nous avons ainsi caractérisé le trafic généré.

Dans un second temps, nous avons simulé un réseau de routeurs RCube soumis à ce trafic, pour vérifier que celui-ci est capable d'acheminer autant de données que les interfaces sont capables de générer et d'absorber.

Nous avons supposé que le trafic était équilibré (hypothèse 2). Si dans la première simulation, nous constatons qu'un composant PCI-DDC est capable d'envoyer et de recevoir une quantité Q de données par unité de temps, et que le réseau simulé dans la deuxième phase est capable, pour N processeurs, de router sans congestion une quantité de données $N * Q$, nous pouvons alors dire que les résultats obtenus dans la première simulation pour deux composants PCI-DDC sont applicables aux N composants PCI-DDC de la deuxième. En sorte, on peut dire que cette décomposition est un « pire cas » : nous regardons d'abord le trafic que les composants

instantané de nombreuses applications, en augmentant leurs performances car il évite les couches réseaux TCP du noyau.

Une étude sur ce point est en cours, sous la direction de Pierre David, au laboratoire PRiSM.

5-3 Conclusion.

Nous avons présenté quatre interfaces logicielles. Celles-ci permettent à des programmes de communiquer des données. On peut les classer par ordre de « connaissance a priori de la structure mémoire de leur partenaire ».

- L'interface « Socket » permet à des applications n'ayant aucune connaissance de leurs structures mémoires respectives de s'échanger des données. Loin du matériel, elle permet de garder la compatibilité avec les applications prévues pour celle-ci. Les échanges de structures de données pour permettre les transferts sont invisibles, ainsi que les interactions avec le gestionnaire de mémoire virtuelle.
- L'interface à adresses virtuelles, ou SR-V, permet d'implémenter un protocole où les applications s'échangent des messages, mais où celles-ci peuvent déjà plus finement gérer les interactions avec le gestionnaire de mémoire virtuelle. En particulier, elles peuvent mettre en œuvre des optimisations sur le verrouillage des zones, de manière à éviter des opérations inutiles de traduction d'adresses successives.
- L'interface à adresses physiques, ou SR-P, est réservée aux programmes qui gèrent eux-mêmes leurs espaces d'adressage physique. Dans notre cas, elle est effectivement réservée au noyau, et plus particulièrement au gestionnaire de mémoire virtuelle qui peut la mettre en œuvre pour gérer de manière globale la mémoire du système. Elle permet de construire, par exemple, un « swap » distribué. Elle peut aussi permettre à des processus, qui ont une connaissance très fine de leurs espaces d'adressage physique respectifs, de communiquer. Après avoir totalement verrouillé leur espace de données, et s'être échangé la topologie de leurs espaces physiques, ceux-ci peuvent communiquer directement avec cette interface.
- L'interface « PUT » est réservée non seulement aux programmes qui ont une connaissance de leur propre mémoire physique, mais aussi de celle de leur interlocuteur. Dans notre système, elle sert à délimiter le travail du logiciel et celui du matériel. Mais on peut fort bien imaginer que celle-ci soit un jour utilisée dans des systèmes d'exploitation autres que UNIX, par des programmes très spécifiques, sans mémoire virtuelle, pour permettre l'utilisation de nos dispositifs dans des machines dédiées au calcul scientifique par exemple.

Un synoptique d'interface logicielle pour les systèmes UNIX permettant l'utilisation de ces mécanismes est proposé. Très modulaire, celui-ci propose quatre niveaux d'abstraction. Suivant les applications et le degré de coopération de celles-ci, l'une ou l'autre de ces interfaces peut être mise en œuvre, avec des performances différentes. Celles de plus bas niveau impliquent que le système soit, dans une certaine mesure, dédié à certaines tâches. L'une d'entre elles, le SR-V, permet un couplage fort entre différentes applications tout en respectant la frontière habituelle entre le noyau et les applications, mais suppose que celles-ci possèdent les droits du super-utilisateur. La dernière, enfin, n'est autre que l'interface « Socket » d'UNIX, pour permettre une utilisation directe de programmes existant sur notre système.

- **int** srv_unlock (**int** *lock_desc*)

  lock_desc est le paramètre retourné par la procédure « Lock ». Il désigne le descripteur de la zone verrouillée. Si le compteur d'utilisation de ce descripteur est non nul, la fonction renvoie un code d'erreur. Sinon, pour chacune des pages, on décrémente le compteur d'utilisation de la page. Si celui-ci est nul, on positionne le drapeau qui permet de déplacer la page dans sa structure associée dans le gestionnaire de mémoire virtuelle.

Les deux routines « Send » et « Receive » de cette interface SR-V sont quasiment identiques à celle de l'interface SR-P.

- **int** srv_receive (**int** *Source*, **int** *VCId*, **pointer** *PR*, **int** *TR*)

  Après avoir trouvé dans la table des descripteurs de zones verrouillées une entrée contenant la zone désignée par (*PR*, *TR*) et incrémenté son compteur d'utilisation, cette routine effectue un appel à srp_receive en passant les emplacements physiques de la zone de mémoire de réception.

- **int** srv_send (**int** *Dest*, **int** *VCId*, **pointer** *PS*, **int** *TS*)

  De même, après avoir effectué les opérations sur le descripteur de zones verrouillées, la routine appelle srp_send.

Cette interface permet directement d'effectuer des opérations de passage de messages entre deux processus. On aurait pu imaginer un système où les appels à « Lock » et « Unlock » n'auraient pas été explicités et induits par des appels à « Send » et « Receive ». Notre séparation présente l'avantage de pouvoir éventuellement économiser des appels inutiles : il est possible d'avoir plusieurs appels aux primitives Send / Receive pour un seul appel aux primitives Lock / Unlock. Dans le cas d'un calcul scientifique, par exemple, où différents processus effectuent des opérations sur de grosses matrices, il est fréquent que des lignes soient échangées. Le système verrouillera donc la totalité de la matrice avant le calcul pour effectuer librement des opérations d'émission et de réception sur des parties et, à la fin du calcul, déverrouillera la mémoire.

Les communications sont sécurisées dans la mesure où l'attribution des zones de mémoire dans lesquelles les émissions et réceptions sont autorisées est faite par la primitive « Lock », qui vérifie les droits des processus sur les zones qu'ils désirent. Ainsi l'altération d'un système par un autre est interdite. Par contre, elles ne sont pas sécurisées dans le sens où un processus, en exécutant des « Receive » avec un identificateur de canal déjà attribué, peut intercepter des communications. Cette interface ne doit donc être utilisée que par le super-utilisateur.

5-2-5 Détournement de l'interface « Socket ».

Une dernière interface consiste à détourner les appels systèmes de l'interface « Socket » du noyau concernant certains numéros IP, et à les rediriger sur notre interface SR-V.

Par l'attribution aux machines présentes sur le réseau HSL d'un deuxième numéro IP, autre que celui du réseau de contrôle, par exemple dans le domaine 10.x.y.z[44] , on peut détourner les trafics effectués sur des sockets vers les composants PCI-DDC.

Ce mode de communication, légèrement dégradé, dans la mesure où par exemple, les « Lock » et « Unlock » sont systématiquement associés aux « Send » et aux « Receive » permet de garder la compatibilité avec les exécutables utilisant les sockets. Il permettra un portage

[44] [RFC1597]

mémoire physique libre, il déplace des pages vers d'autres machines du réseau HSL. Au vu des vitesses comparées des disques actuels et du réseau HSL, on peut espérer une amélioration d'un facteur 100 du coût d'un défaut de page avec un tel système, par rapport au déplacement des pages vers le disque.

5-2-4 Interface « SR-V ».

Cette interface permet d'effectuer des transferts sur un modèle par passage de messages, en encapsulant les interactions avec le gestionnaire de mémoire virtuelle. Elle possède deux primitives supplémentaires qui réalisent la traduction d'adresse :

- **int** srv_lock (**pointer** *P*, **int** *T*)

  P et *T* désignent une zone de mémoire virtuelle. Dans le cas général, celle-ci est représentée sur la figure 5-e. La fonction « Lock » effectue tout d'abord les opérations suivantes pour chaque page de la zone concernée :

  1. Si la page est dans le « swap » du système, on la ramène en mémoire physique.
  2. Dans le descripteur de la page géré par le gestionnaire de mémoire virtuelle, on positionne le drapeau qui indique que la page ne peut être déplacée, puis on incrémente un compteur d'utilisation de la page.
  3. On calcule l'adresse physique de la table.

  Ensuite, on construit un descripteur, dans la table des descripteurs de zones verrouillées, qui contient les informations suivantes :

  - *P*, *T* décrivant la zone en mémoire virtuelle.
  - P1, ... Pn contenant les adresses physiques des pages qui contiennent cette zone.
  - Un compteur d'utilisation du descripteur.

Ce compteur d'utilisation sert à connaître le nombre de « Send » et de « Receive » qui sont en cours sur cette zone, et à éviter ainsi de libérer des zones qui sont en cours d'utilisation.

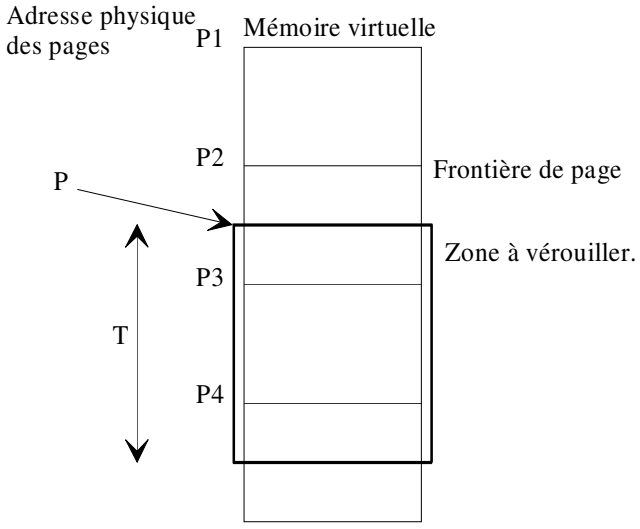

Figure 5-e : Verrouillage.

La primitive symétrique de la précédente, « Unlock » , s'effectue de la manière suivante :

D'autre part, les processeurs gardent une trace des émissions et des réceptions qu'ils ont en cours. Pour cela ils maintiennent les tables « pending-send » et « pending-receive ».

Le processeur émetteur peut donc effectuer un « Direct Deposit » des données dans la zone de réception, puisque sa structure lui a été préalablement communiquée lors du rendez-vous.

- **int** srp_receive (**int** *Source*, **int** *VCId*,
  pointer *PR1*, **int** *TR1*, ..., **pointer** *PRn*, **int** *TRn*)

  Cette primitive effectue les opérations suivantes :

  1. Ecriture dans la table locale des « pending-receive » pour conserver une trace du « Receive » en cours.
  2. Allocation d'un identificateur de message « IM » qui va servir à marquer tous les paquets associés au transfert effectif des données.
  3. Envoi d'un message, qui écrit dans la table « Received Receive » de la machine *Source* la liste des pages constituant la zone mémoire de réception et le « IM » choisi. La réception de ces informations génère une interruption sur la machine émettrice du message. On note dans « Bitmap Received Receive » les emplacements pris dans cette table.

- **int** srp_send (**int** *Dest*, **int** *VCId*,
  pointer *PS1*, **int** *TS1*, ..., **pointer** *PSn*, **int** *TSn*)

  1. Ecriture dans la table locale des « pending-send » des paramètres du « Send » effectué.
  2. Vérification dans la table locale des « Received Receive » associée à la machine *Dest* de la présence d'une entrée portant l'identificateur de canal *VCId*.
  3. Dans le cas où cette entrée n'est pas présente, la primitive se termine et l'envoi des données sera effectué lors de l'interruption déclenchée par la primitive « Receive ». Dans le cas où celle-ci est présente, l'envoi effectif du message est effectué.

Une procédure supplémentaire gère les interruptions envoyées par l'interface PCI-DDC. Celles-ci peuvent être de plusieurs types :

- Interruption de réception de messages : un message vient d'être reçu. Deux cas sont possibles. Soit ce message est un message de signalisation « Received Receive ». Dans ce cas, on regarde dans la table des « Pending-send » si une entrée portant le même identificateur de canal est présente pour effectuer l'envoi effectif des données. Soit le message reçu est un transfert de données, dans ce cas on détruit l'entrée dans la table « Pending-receive » associée, et on libère dans « Bitmap Received Receive » les entrées associées à l'envoi préalable du descripteur de la zone de réception.
- Interruption de fin d'envoi de message : un message vient d'être totalement émis. On détruit l'entrée associée dans « Pending-send ».

Les détails du fonctionnement de ce protocole peuvent être trouvés dans l'annexe 4. On peut déjà faire les remarques suivantes :

- Ce mécanisme permet, avec la seule primitive d'écriture à distance de générer des transferts de type passage de messages.
- Son « Overhead » est assez faible. Il s'agit de faire correspondre les identificateurs de canaux dans le cas d'un « Send » et dans le cas de réception d'un message « Receive ».

Cette interface qui manipule des adresses physiques peut être utilisée, par exemple, par un système de « swap distribué ». Le principe est que lorsqu'un système a besoin d'espace

Un exemple d'utilisation : MPC.

5-2-3 Interface « SR-P ».

Cette interface utilise des adresses physiques. Elle possède deux appels principaux « Send » et « Receive », qui permettent d'effectuer des transferts dans un modèle par passage de messages classique. On suppose que les zones de mémoire d'émission et de réception ont été préalablement déclarées aux gestionnaires de mémoire virtuelle pour qu'ils assurent qu'elles ne seront pas déplacées ou mises sur le disque pendant le temps des transferts. D'autre part, cette couche logicielle gère quatre tables supplémentaires, qui maintiennent des informations sur les transferts en cours. La figure 5-d présente le synoptique d'un transfert complet.

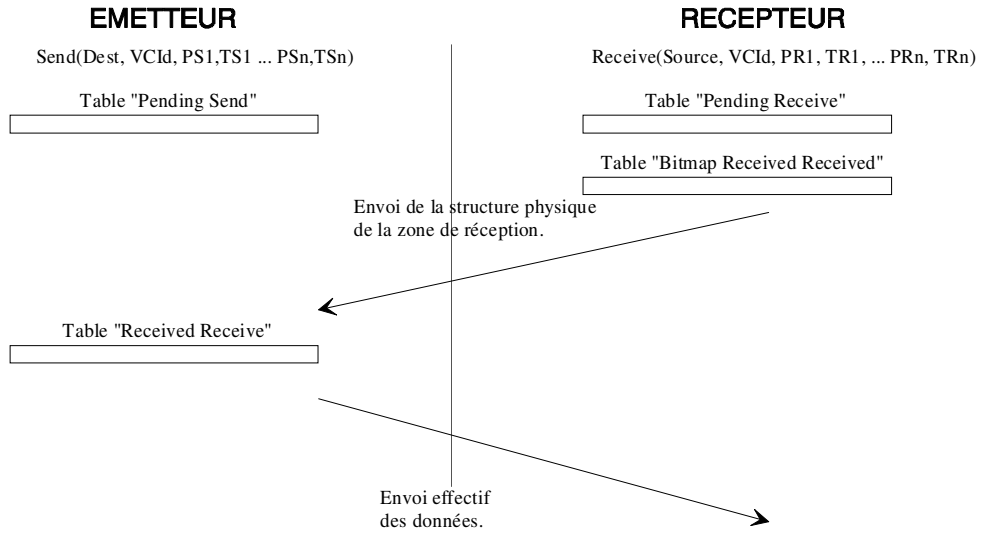

Figure 5-d : synoptique d'un transfert complet.

Le « rendez-vous » entre deux processus est effectué grâce à un identificateur de canal, VCId, complété par un numéro de séquence. Pour plus de détail sur la gestion de ces identificateurs, on peut se référer à l'annexe 4.

Les deux processus qui désirent effectuer des transferts réalisent des appels aux fonctions « Send » et « Receive », avec ce même identificateur. La manière dont celui-ci est obtenu est spécifiée par les couches logicielles supérieures. On peut se référer à l'annexe 4 pour plus de détails.

Les tables « Received Receive » jouent un rôle particulier dans notre système : elles correspondent à des « boîtes aux lettres » et sont utilisées pour initier un transfert. Chaque processeur possède (N-1) tables « Received Receive » associées aux (N-1) autres processeurs du système. Elles sont écrites par la machine qui effectue le « Receive » et lues par celle qui effectue le « Send ». Elles sont allouées au démarrage du système dans des zones de mémoire du noyau et leurs adresses physiques respectives sont communiquées à tous les processeurs par le réseau de contrôle.

Une entrée dans ces tables est associée à chaque transfert de messages. Le processeur récepteur du message y écrit préalablement la structure de sa zone de réception. Cette écriture constitue le rendez-vous préalable au transfert.

Grâce à ces tables « Received Receive », le processeur émetteur a accès aux adresses physiques de la zone de réception des messages chez les processeurs récepteurs. Le récepteur garde dans une table « Bitmap Received Receive » l'image des entrées actives des tables « Received Receive » dans les processeurs distants.

du noyau, en particulier des gestionnaires d'entrée-sortie répartis, ou bien une application de « swap » mémoire distribuée.

- Une troisième interface, « PUT », reflète au niveau logiciel notre primitive matérielle d'écriture à distance (« Remote Write »). Celle-ci peut être aussi utilisée par d'autres parties du noyau, dans des cas où des applications ayant une connaissance fine de leurs mémoires respectives cherchent à coopérer.

Nous allons successivement décrire ces trois interfaces, en partant de celle de plus bas niveau, puis en remontant vers l'abstraction la plus élevée. Plus cette interface s'éloigne du matériel, plus le coût est élevé du point de vue de la performance, mais plus le portage des applications existantes est aisé.

5-2-2 Interface « PUT ».

Le but de cette interface est de gérer le composant PCI-DDC et ses structures de données propres et de proposer au programmeur un moyen de s'affranchir de la gestion des registres du composant, ainsi que de tout l'aspect purement matériel.

Cette interface ne comporte que quatre primitives. La principale est « PUT », dont la syntaxe est la suivante.

- **int** pciddc_put (**int** *Remote_Node*, **int** *Message_Identifier*,
  pointer *Local_Address*, **pointer** *Remote_Address*, **int** *Size*)

  Cette fonction permet de rajouter une entrée dans la Liste des Pages à Emettre (LPE) et d'en informer le coposant PCI-DDC. Les paramètres *Remote_Node* et *Message_Identifier* sont ceux présents dans une entrée de la LPE décrite au chapitre 4. Les deux adresses désignent des emplacements physiques respectivement dans la mémoire locale et la mémoire distante. Cette primitive est donc le reflet direct de notre principe d'écriture à distance.

Trois autres primitives sont disponibles :

- **int** pciddc_init

  Cette primitive vérifie la présence d'un composant PCI-DDC dans le système, crée les structures de données qui lui sont associées (LPE, LMR), initialise le composant en positionnant ses différents registres et enfin active les communications. Elle renvoie un code d'erreur dans le cas où ces opérations n'ont pu être effectuées correctement.

- **int** pciddc_shutdown

  Cette procédure, après avoir interdit au composant PCI-DDC l'émission et la réception de paquets, détruit les structures de données qui lui sont associées. Celle-ci permet d'effectuer des redémarrages à chaud du système, dans les cas où une erreur grave a été détectée.

- **int** pciddc_setcallback

  Cette routine permet de définir la routine à appeler pour gérer les interruptions émises par le composant PCI-DDC. Lors de son exécution, les paramètres codant le type d'interruption, ainsi que d'éventuelles données, lui sont transmis.

Seule la première de ces primitives, « PUT », est utilisée communément, les autres ne servant qu'au démarrage et à l'arrêt de la machine.

Un exemple d'utilisation : MPC.

5-2 L'interface logicielle Send / Receive.

5-2-1 Présentation.

L'ambition de cette étude est de présenter les mécanismes fondamentaux d'une interface logicielle mettant en œuvre notre réseau d'interconnexion et de montrer qu'ils peuvent être ajoutés à un système d'exploitation de type UNIX. La présentation de cette interface sert essentiellement, au sein de ce manuscrit, à permettre la compréhension des mécanismes matériels, et à montrer qu'il est possible d'effectuer des envois et des réceptions de messages à l'aide de la seule primitive d'écriture à distance. Pour plus de détails, on peut se référer à sa spécification plus complète, rédigée par Alexandre Fenyö, dans l'annexe 4. Il s'agit en fait de « parer au plus pressé » et de proposer un mécanisme de transfert de messages le plus rapidement possible. Des implémentations plus fines d'interfaces logicielles mieux adaptées aux besoins des machines parallèles feront l'objet d'études ultérieures.

Sur la figure 5-c, nous représentons une vue générale de l'architecture logicielle.

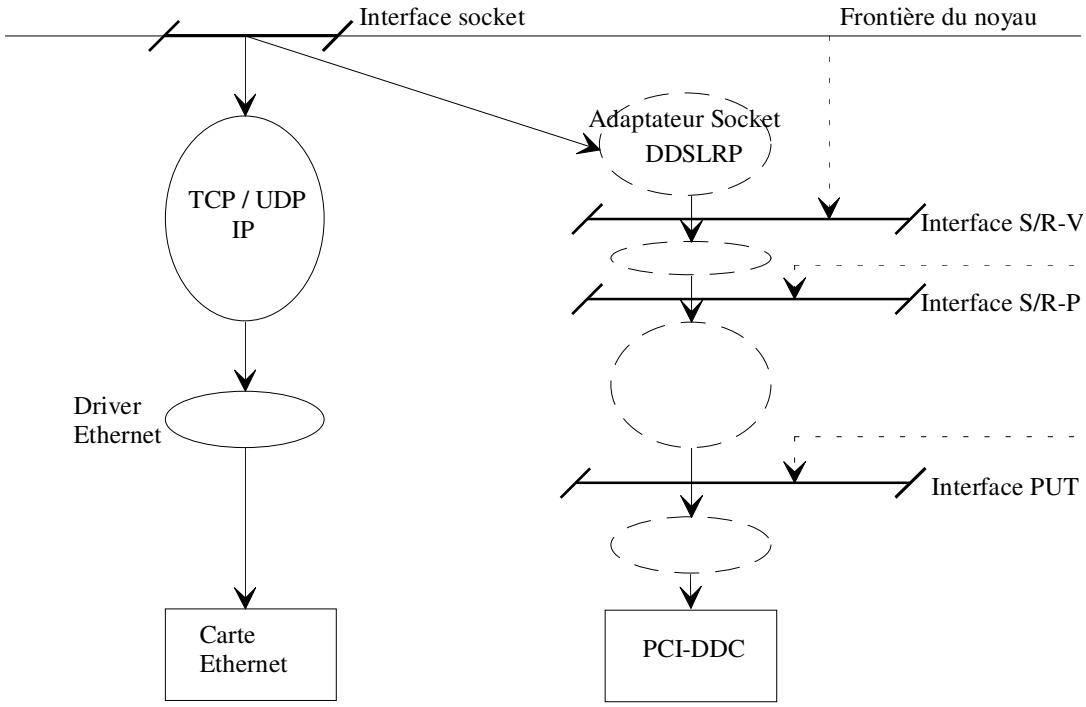

Figure 5-c : architecture globale de l'interface logicielle.

- Sur la gauche, l'interface classique des transferts utilisée par les stations UNIX, TCP / UDP et IP est représentée.
- Parallèlement à celle-ci, une interface logicielle SR-V (Send & Receive on Virtual memory) est disponible. Elle peut éventuellement être atteinte directement par des processus applicatifs qui disposent des droits du super-utilisateur. Cette première interface, décrite en détail dans le paragraphe 5-2-4, utilise notre système de communication. Elle règle les problèmes de traduction d'adresse virtuelle / physique et assure la synchronisation et la vérification des droits des processus sur la mémoire qu'ils utilisent.
- Une seconde interface SR-P (pour mémoire Physique), de plus bas niveau, implémente uniquement la synchronisation des processus émetteurs et récepteurs. Elle suppose que la traduction des adresses a déjà été effectuée. Celle-ci peut être utilisée par d'autres éléments

composant d'interface, mais aussi un routeur à chaque cellule processeur. Cette association d'un routeur par processeur permet, entre autre, de faciliter la configuration des routeurs, comme nous avons pu le voir au paragraphe 4-3-5, et de détecter d'éventuelles anomalies beaucoup plus simplement que si les routeurs étaient regroupés sur une carte séparée.

Notre routeur présente une valence de 8 liens. L'un d'eux est dirigé vers le composant d'interface. Chaque nœud présente donc une valence de 7 liens de communications, ce qui nous permet de construire aisément des topologies régulières pour des systèmes jusque 256 processeurs et plus.

Parallèlement à ce réseau à haut débit, chaque carte processeur possède une interface Ethernet, que l'on utilise comme réseau de maintenance et de configuration. Celui-ci permet la mise au point du système, le diagnostic d'éventuels dysfonctionnements du réseau haut débit et la reconfiguration de manière simple.

5-1-3 Architecture.

Nous présentons de manière synthétique l'architecture de notre machine sur la figure 5-b. D'un point de vue matériel, cette architecture nécessite le développement d'une seule carte d'extension comportant un routeur RCube et un circuit PCI-DDC.

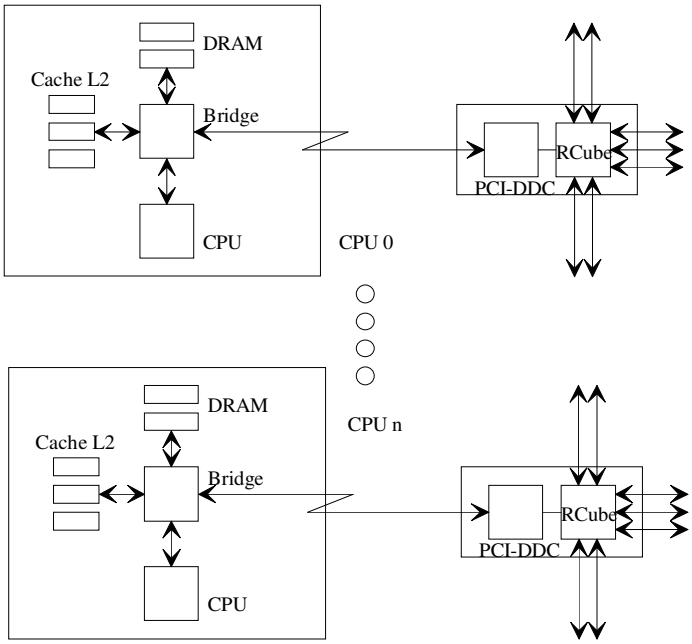

Figure 5-b : présentation synthétique de notre architecture.

Les techniques de routage explicite ou de routage par intervalles permettent de générer des topologies très diverses pour une machine avec différents nombres de processeurs. Avec 7 liens, pour 8 processeurs, une topologie pleinement interconnectée est possible. Pour 128 processeurs, l'hypercube semble une bonne solution. Pour d'autres nombres, y compris ceux qui ne sont pas des puissances de 2, de multiples topologies sont possibles[43]. Nous en présentons une dans le chapitre 6.

En plus de ce réseau d'interconnexion, chaque noeud comporte une carte Ethernet. Celles-ci forment un réseau de contrôle, qui sert à la configuration et au démarrage du système.

[43] [BOUARA]

direction. On citera notamment les projets DASH[37] et FLASH[38] de Stanford et Alewife[39] du MIT

- Les machines à mémoire virtuelle distribuée et partagée (DVSM) : chaque processeur possède sa propre mémoire, avec son espace d'adressage propre, par contre la mémoire virtuelle, vue par les processus, est partagée, par réplication ou déplacement de pages de mémoire physique. Ici, les développements matériels sont plus limités, car un grand nombre d'opérations sont effectuées par le logiciel, mais il semble alors que les performances de ce genre de machines sont limitées par la lourdeur de ces mécanismes logiciels. Le projet SCHRIMP[40] de Princeton utilise des mécanismes de ce type.
- Les machines à passage de messages (MPA) : chaque processeur possède sa propre mémoire et son propre espace d'adressage, et il communique en échangeant des messages, qui sont recopiés d'une mémoire à l'autre. Là aussi, les développements matériels sont limités. Souvent, ce genre de machine implique que le programmeur, ou le compilateur, ait explicité les transferts de données entre les différents éléments concurrents de son programme, ce qui est un exercice difficile. Le CM-5[41] est certainement l'exemple le plus connu.
- Les réseaux de stations de travail (LAN) : différents ordinateurs coopèrent à l'exécution de tâches en utilisant les outils de communication pour réseaux locaux. Aucun développement matériel spécifique n'est demandé car on utilise les dispositifs des réseaux locaux et des bibliothèques de programmation du type PVM ou MPI, les performances de ce genre de système sont gravement limitées par les faibles débits et les mécanismes logiciels complexes des protocoles de réseaux locaux.

Nous visons une machine à passage de messages (MPA).

Matériellement, notre machine parallèle peut se présenter sous deux formes différentes. Tout d'abord, un cabinet contenant l'ensemble des cartes processeurs du système, ainsi que le réseau d'interconnexion et les alimentations électriques peut être développé. Le système est alors relié par un brin Ethernet au réseau local.

Une autre solution consiste à prendre une salle de stations de travail, et à rajouter à chacune d'entre elles notre système d'interconnexion, en plus des connexions Ethernet. Cette seconde approche présente bien des avantages, surtout dans un cadre universitaire, car elle permet la mise au point des matériels et des logiciels dans des conditions moins coûteuses que la première approche. Cette approche est proposée par D. Patterson, et évoquée dans le chapitre 1[42] .

5-1-2 Le réseau d'interconnexion.

Le choix de la liaison point-à-point, ainsi que notre volonté d'avoir un système extensible, ont impliqué que notre réseau d'interconnexion présente un composant de routage de paquet, RCube, qui est présenté dans le chapitre 3.

Le composant, PCI-DDC (voir chapitre 4), assure l'interface entre le bus PCI et le lien HSL. Nous avons choisi de distribuer la fonction de routage : on associe non seulement un

37 [LENOSKI] et [KARAM]
38 [HE&Al-1] et [HE&Al-2]
39 [CHAIKEN94] et [CHAIKEN95]
40 [BLUMRICH].
41 [LEISER] et [LIN]
42 [PATTER94] et [PATTER95]

Chapitre 5 : Un exemple d'utilisation : MPC.

La mise à disposition prochaine des composants RCube et PCI-DDC va permettre d'effectuer des recherches sur les mécanismes logiciels qui permettent de reporter jusqu'à l'application le maximum du débit proposé par le réseau d'interconnexion, tout en conservant une latence minimale.

Pour permettre ces recherches, nous proposons la construction d'une plate-forme de développement intégrant le réseau présenté dans les chapitres 3 et 4 : la machine MPC, qui est présentée ici.

Ce chapitre se compose de deux parties. Dans un premier temps, nous présentons l'utilisation que nous faisons des deux composants RCube et PCI-DDC, puis un premier exemple d'interface logicielle de mise en œuvre de ce système d'interconnexion est esquissé.

5-1 La machine MPC.

5-1-1 Positionnement.

On peut classer les architectures MIMD par degré de couplage, c'est-à-dire par le degré d'interaction des différents processeurs entre eux. La figure 5-a présente un classement dans lequel on peut distinguer plusieurs grandes familles. Ce classement reflète les différents mécanismes matériels présents dans ces architectures pour supporter le parallélisme.

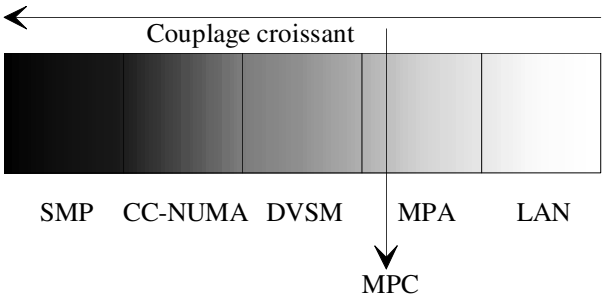

Figure 5-a : *taxinomie des machines parallèles.*

- Les grappes de processeurs (SMP pour Symetric Multi Processor) : plusieurs processeurs partagent une mémoire commune, centralisée. Les processeurs sont en général sur la même carte et accèdent à la mémoire au moyen d'un bus commun. Ces machines exigent des développements matériels importants, en particulier la modification des contrôleurs d'interruption.
- Les machines à accès non-uniforme et cohérence de cache (CC-NUMA) : la mémoire est physiquement répartie sur chaque carte processeur, mais les processeurs partagent logiquement l'ensemble de la mémoire physique. Il n'y a qu'un seul espace d'adressage physique. Ici, des composants électroniques doivent être développés pour transformer les accès mémoire d'un processeur en requêtes dirigées vers les autres, et pour gérer la cohérence des caches. De nombreuses équipes effectuent des recherches dans cette

- Aucune vérification des droits d’écriture et de lecture en mémoire n’est effectuée par le matériel. Elles sont à la charge du système d’exploitation. Celui-ci doit en particulier veiller à ce que des transferts intempestifs n’altèrent pas l’intégrité des systèmes distants.
- La traduction des adresses virtuelles en adresses physiques est également à la charge du système d’exploitation.
- L’utilisation d’appels système est obligatoire, autant pour effectuer les vérifications de droits en écriture et lecture que pour accéder aux ressources partagées utilisées par le contrôleur réseau.

Le composant PCI-DDC implémente notre protocole « Direct Deposit Stateless Receiver Protocol ». L’utilisation des mêmes opérateurs que RCube pour gérer l’asynchronisme des horloges permet d’émettre et de recevoir des données avec des horloges indépendantes de celle du bus PCI sur lequel il se trouve. Il implémente une gestion de flux avancée, qui lui permet d’être utilisé avec d’autres composants IEEE 1355. Il présente une mémorisation plus importante que RCube, pour atténuer le fait que le bus PCI soit monodirectionnel, alors que le lien HSL est full-duplex. Il permet le transfert de données, y compris dans les cas de non-alignement les plus généraux, sans mettre en œuvre de technique lourde, mais en utilisant le fait que le lien HSL manipule des données alignées sur un octet. Son automate de contrôle est unique. Ceci permet de ne pas avoir à gérer la coopération de différents automates dans le même composant, et donc de simplifier sa mise au point. D’autre part, le composant PCI-DDC inclut les opérateurs permettant la configuration d’un RCube qui lui est associé, depuis le bus PCI, c’est-à-dire par logiciel.

Un premier modèle VHDL de niveau RTL du composant PCI-DDC a été écrit. Ils a permis d’évaluer de façon réaliste la complexité matérielle du circuit. La partie contrôle de cette première étude comportait deux automates communicant, ce qui introduisait une complexité inutile. Jean-Lou Desbarbieux et Frank Wajsbürt étudient actuellement une seconde version avec un seul automate, telle qu’elle est présentée dans le paragraphe 4-3-6. La réalisation du modèle VHDL n’a donné lieu à aucune difficulté micro-électronique. Il apparaît que la réalisation de notre composant, intégrant des dispositifs de communication performants, peut se faire tout en gardant une faible complexité matérielle.

the most frequent case in Hardware » reste notre mot d'ordre. Ainsi la détection par le matériel de ces erreurs entraîne une interruption logicielle. Pour cela, 4 mécanismes complémentaires sont mis en œuvre :

- Sur une liaison point-à-point, chaque octet possède son propre bit de parité, ce qui permet d'effectuer une première vérification des données.
- L'en-tête du paquet possède son propre code redondant. Celui-ci permet de détecter le cas où deux bits d'erreur au sein du même octet ont trompé le mécanisme de détection de parité. Comme il est extrêmement grave qu'une erreur se produise sur l'en-tête (adresse erronée), le paquet sera détruit.
- Un troisième code redondant permet de vérifier l'intégrité des données. En effet, celles-ci peuvent contenir des adresses physiques et donc, par ricochet, générer de graves erreurs systèmes en cas d'altération. La détection d'erreur se fait après l'écriture du paquet en mémoire.
- Un quatrième phénomène particulièrement grave peut se produire : la fin d'un paquet n'est pas détectée, et ainsi, le paquet suivant celui qui a été altéré vient, s'écrire en mémoire, dépassant ainsi la zone réservée à la réception. La probabilité d'apparition de ce phénomène est diminuée par la redondance des caractères ED et EP. Si le composant PCI-DDC détecte un seul de ces deux caractères sans l'autre, la réception s'interrompt.

Le taux d'erreur observé sur le lien HSL est extrêmement faible. Il est du même ordre de grandeur que celui observé sur un bus de fond de panier. Le caractère extrêmement grave des erreurs dans notre protocole, qui peuvent conduire à la destruction d'un système distant par une machine du réseau, ainsi que la volonté d'avoir une forte disponibilité pour une machine avec un grand nombre de processeurs, nous ont conduit à la définition de ces codes redondants.

Leur « coût » en matière de bande passante et « d'Overhead » est assez faible : 9 octets par paquet.

4-4 Conclusion.

Nous venons de présenter le composant d'interface de notre réseau d'interconnexion. Il réalise une seule primitive, qui est l'écriture à distance. Il permet toujours d'utiliser des techniques classiques de communication, mais permet, dans les cas où cela est possible, la suppression des recopies à la charge du processeur. D'autre part, il présente les avantages suivants :

- La reconstruction des paquets en messages s'effectue de manière automatique.
- Une seule interruption est nécessaire par message, et non plus une par paquet.
- Le transfert des données peut s'effectuer sans aucune recopie, grâce au Direct Deposit Protocol.
- D'autre part, ce transfert s'effectue sans intervention du processeur principal, par DMA.
- Les limitations implicites de taille de mémoires tampon de réception disparaissent grâce à un rendez-vous préalable au transfert effectif des données.
- Le nombre d'accès entrée-sortie est limité, au profit d'accès mémoire, plus performants.

Il présente cependant un certain nombre d'inconvénients :

Dans certains états d'émission ou de réception, où le composant PCI-DDC demande le bus PCI, des accès de configuration peuvent apparaître. Dans ce cas, le composant demande le bus en même temps que le bridge CPU-PCI. Ce dernier l'obtient et effectue des accès de configuration. Notre automate est donc un automate à pile. Dans les états où notre composant n'est pas maître du bus, on teste la présence d'un accès de configuration, et on déclenche les opérations associées à celui-ci.

Notre automate d'état se présente donc sous la forme d'un papillon, représenté sur la figure 4-r.

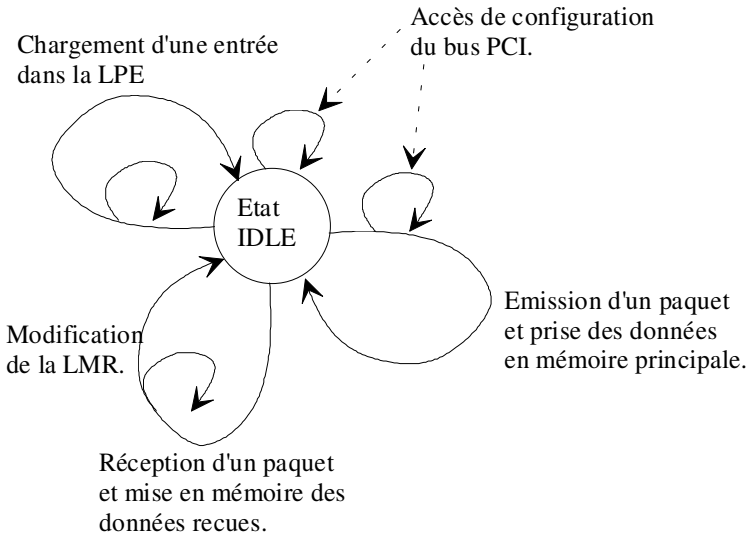

Figure 4-r : automate principal de PCI-DDC.

Ceci permet l'écriture d'un automate simple, avec, certes, un nombre important d'états, mais avec peu de transitions entre eux. Les séquences de configuration, qui ne possèdent que quatre états au maximum ne sont pas répétées, mais incluses sous la forme d'un « appel » à un sous microprogramme.

D'une manière plus fine, l'automate va aussi effectuer un choix important : faut-il favoriser l'émission ou la réception des données ? En effet, suivant la priorité avec laquelle les deux branches de réception et d'émission sont codées dans l'état « IDLE » de l'automate, soit les réceptions, soit les émissions vont être favorisées. On voit en détail cet aspect dans le chapitre 6, mais l'on peut d'ores et déjà dire que cela dépend des topologies et du type de trafic mis en œuvre par les logiciels. Bref, cette priorité doit être configurable, grâce au registre « Interface Mode ». L'expérimentation permettra de faire le meilleur choix.

Nous avons aussi fait le choix de synthétiser un automate câblé, plutôt que de le microprogrammer. Ceci simplifie grandement son implémentation et augmente sa rapidité. Une solution micro-codée, avec le micro-code chargé lors du démarrage du composant, présente cependant des avantages certains. On modifie le micro-code dans le cas d'erreur sur l'automate, et on configure des modes de priorités plus fins entre la réception et l'émission. Mais sa plus grande complexité nous a fait opter pour une solution d'automate synthétisé.

4-3-7 Gestion des erreurs.

Un autre rôle de PCI-DDC est de vérifier l'intégrité des paquets reçus. Nous nous bornons en effet, comme dans le routeur RCube, à détecter les erreurs, et ainsi à éviter des écritures illégales en mémoire. Aucun traitement de récupération des erreurs n'est effectué. « Support

- « LMR Descriptor » : un registre sert à donner l'adresse physique de la liste des messages en réception. Ce registre est écrit au démarrage du système.

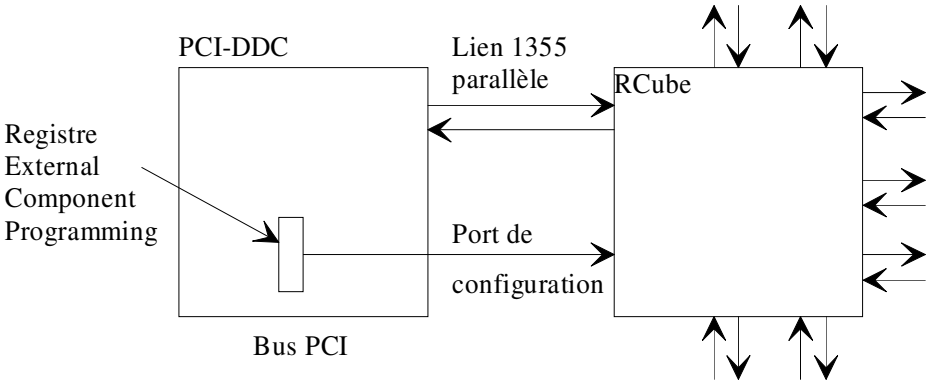

Figure 4-q : chargement des tables de RCube par PCI-DDC.

4-3-6 L'automate de contrôle.

Nous avons présenté les automates qui contrôlent les files d'attente vers le réseau HSL. Ceux-ci fonctionnent totalement indépendamment du reste du composant. La réception des données, leur émission et le contrôle de flux y sont gérés.

L'interface que proposent ces files d'attente au reste du composant est donc une interface classique, avec deux fonctions : « PUSH » pour mettre une donnée dans la file d'attente d'émission et « POP » pour retirer une donnée depuis la file d'attente de réception.

Le « Direct Deposit Stateless Receiver Protocol » effectué par le composant PCI-DDC est un mécanisme simple. Dans le composant, un automate d'état gère ce protocole. Il pilote trois parties opératives associées à chacune des trois tâches du composant : la réception de paquet, l'émission, et les accès de configuration du bus PCI.

En première approche, on pourrait penser que deux automates coopèrent. L'un pour gérer le bus PCI, c'est-à-dire servir les accès de configurations et effectuer les accès en mode maître, l'autre pour gérer la mise en forme et l'analyse des paquets. On peut cependant remarquer que, le bus PCI étant soit en lecture soit en écriture, le composant PCI-DDC n'effectue jamais qu'une seule chose à la fois, donc cette concurrence de deux automates distincts n'est qu'une source de complexité. En effet, notre interface peut, et de manière exclusive, n'effectuer que l'une des tâches suivantes :

- Charger une entrée depuis la Liste des Pages à Emettre : la réception et l'émission des données sont effectuées dans les files d'attente, en parallèle avec cette lecture.
- Emettre un paquet vers le réseau HSL : le composant écrit dans la file d'attente d'émission et est prévenu lorsque celle-ci arrive à saturation. La réception se fait parallèlement dans la file d'attente prévue à cet effet.
- Ecrire en mémoire un paquet reçu: les données présentes dans la file d'attente d'émission continuent à partir sur le lien HSL, tandis que la réception de données s'effectue parallèlement. Ensuite, après chaque paquet reçu, il faut mettre à jour la « liste des messages en réception ».
- Servir des accès de configuration : ceux-ci ne peuvent se produire que lorsque le composant ne contrôle pas le bus.

Fin de paquet Erreur	Byte Enable					Octet 0	Octet 1	Octet 2	Octet 3
0	1	1	1	1	1	NPR			EP
0	1	1	1	1	1	CRC			
0	0	1	1	1	0	Mot 3			
0	0	1	1	1	1	Mot 2			
0	0	0	0	1	1			Mot 1	
0	0	1	1	1	1	CRC			
0	0	1	1	1	1	Adresse			xxxx10
0	0	1	1	1	1	Dest		IM	

Figure 4-p : file d'attente de réception et alignement.

4-3-5 Protocole PCI.

La norme PCI impose que notre composant présente un mécanisme, dit de configuration, permettant au processeur de lire ou d'écrire un certain nombre de registres. Ceux-ci se divisent en deux catégories, les registres standards, contenant des informations sur le type de périphérique, son constructeur etc. présents dans tous les composants PCI, et d'autres dont l'utilisation est laissée libre.

Nous avons décidé que les registres utilisés pour le protocole de communication se trouveraient dans cette zone de configuration. Ceci nous évite de devoir mettre en œuvre, parallèlement à ce premier mécanisme, un second, dans lequel nos registres se trouveraient dans la zone d'entrée-sortie.

On peut énumérer rapidement ces 7 registres. Pour plus de détails, on peut se référer à la spécification du composant dans l'annexe 3.

- « Interface Command » : ce premier registre contient des commandes permettant d'activer ou désactiver les diverses fonctions du composant. On peut ainsi n'autoriser sélectivement que la réception ou l'émission, désactiver la fonction de détection d'erreurs, et déterminer les conditions qui doivent produire des interruptions.
- « Interface Status » : on peut, en interrogeant ce registre, connaître l'événement qui a produit une interruption (émission, réception, erreur etc.).
- « External Component Programming » : les valeurs contenues dans ce registre sont intégralement reportées sur des plots de PCI-DDC. Celui-ci permet de contrôler le scan-path de configuration d'un RCube, par exemple (figure 4-q).
- « Received Message » : ce registre contient l'identificateur « IM » du dernier message reçu. Dans le cas d'une interruption de réception, il n'est donc pas nécessaire de parcourir la table des identificateurs de messages pour savoir lequel d'entre eux est arrivé.
- « LPE Descriptor » : quatre registres servent à décrire la LPE. Ils contiennent respectivement l'adresse de début dans la mémoire physique de la machine, l'adresse de fin, le pointeur de la page à émettre et le pointeur de dernière page ajoutée. Ces registres sont positionnés au démarrage du système.

- Puis il effectue un accès PCI maître sur les mots contenant les données à transmettre. Il écrit ces mots dans la file d'attente de sortie, en positionnant les « Byte Enable » calculés à partir de l'adresse physique locale et de la taille du paquet qui lui ont été fournies par le logiciel.
- Il pousse ensuite un caractère de contrôle spécial, caractère « ED » pour « End of Data », choisi parmi les caractères de contrôle libres de la norme IEEE 1355, puis la queue du paquet.

9eme bit	Byte enable				Octet 0	Octet 1	Octet 2	Octet 3
0	1	1	1	1	Destination		IM	
0	1	1	1	1	Adresse distante			xxxx10
0	1	1	1	1	CRC			
0	0	1	1	1		Mot 1		
0	1	1	1	1	Mot 2			
0	1	1	0	0	Mot 3			
1	0	0	0	1				ED
0	1	1	1	1	CRC			
1	1	1	1	1	NPR			EP

Figure 4-o : file d'attente d'émission et alignement.

Le bloc de conversion 37 bits vers 8 bits envoie ensuite les octets présents dans la file d'attente sur le lien HSL. Ceux auxquels est associé un « Byte Enable » à zéro se voient remplacés par un caractère « IDLE ». Le paquet ainsi envoyé sur le réseau n'est composé que d'une suite d'octets que le récepteur va réaligner, à partir des deux derniers bits de l'adresse distante. Un certain nombre de « trous », ou caractères « IDLE » sont insérés dans le paquet, mais ce dernier est compacté lors de son routage à travers les RCube, comme nous l'avons vu dans la section 3-3-2.

Réception.

A la réception d'un paquet, le bloc de conversion 8 bits vers 32 bits va réaligner les données du message, en tenant compte des deux derniers bits de l'adresse qu'il reçoit. La file d'attente de réception a une largeur de 38 bits. En effet, deux bits supplémentaires, « Erreur de parité » et « Fin de paquet » ont été ajoutés. Ils sont générés par ce même bloc de conversion. Le caractère « Fin de paquet » est positionné à un après la réception du mot de contrôle « ED », c'est-à-dire sur les deux derniers mots du paquet.

Le bloc de conversion assure le réalignement des données en analysant les deux derniers bits de l'adresse. Comme la fréquence de fonctionnement de ce bloc est élevée (80 Mhz), on minimise ainsi les traitements qui y sont effectués.

Le mécanisme présenté permet de gérer les inconvénients liés au désalignement des données en émission et en réception dans le cas le plus général, en minimisant la complexité des traitements effectués à fréquence élevée. Il n'introduit qu'une légère perte de bande passante, au maximum 7 octets, dont 6 sont des caractères « IDLE » dont le surcoût disparaît au cours du routage.

d'attente asynchrone. Un compteur à trois positions est aussi tenu, incrémenté par la réception de ce signal « FCC Received », décrémenté par celle de la recopie d'un « flit » complet dans la file d'attente asynchrone depuis la file d'attente synchrone. Ces recopies n'ont lieu que si le compteur est non nul, indiquant ainsi un crédit d'émission.

Grâce à ces deux mécanismes, PCI-DDC peut cohabiter avec n'importe quel composant implémentant la norme IEEE 1355 et possédant un compteur de crédit d'au moins deux flits.

Comme les blocs s'échangeant ces signaux de contrôle de flux fonctionnent à des horloges différentes, un mécanisme de resynchronisation de signaux de contrôle a été développé. Il s'agit du même que celui qui est utilisé dans RCube pour la resynchronisation des signaux de contrôle de flux.

4-3-4 Gestion de l'alignement.

Notre mécanisme de déplacement direct en mémoire des données nous pose un autre problème, celui de la gestion du désalignement des données entre la mémoire du récepteur et celle de l'émetteur. On peut en voir le cas général sur la figure 4-n.

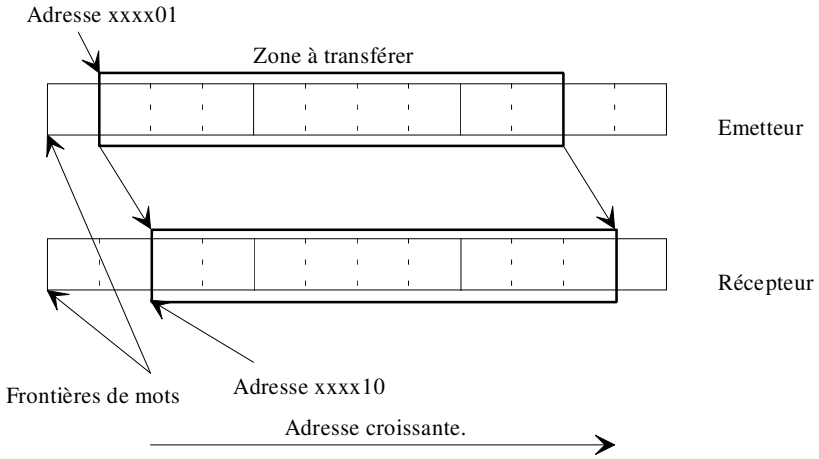

Figure 4-n : cas général de désalignement émetteur / récepteur.

Emission.

Le bus PCI ne permet de faire des opérations de lecture et d'écriture, en mode maître, que sur des mots alignés. Cependant, il est possible, grâce à quatre bits « Byte Enable » de sélectionner une partie des mots. Dans ces files d'attente, nous rajoutons donc quatre bits, désignant, aussi bien en émission qu'en réception, les parties du mot présent dans la file d'attente qui contiennent effectivement des données. La figure 4-n présente, pour l'exemple ci-dessus, les valeurs inscrites dans la file d'attente d'émission.

Aux 36 bits (32+4) de la file d'attente, nous ajoutons un 37 ème bit qui correspond au 9 ème bit de l'octet de poids faible du mot de 32 bits. Il permet ainsi d'envoyer des caractères de contrôle sur le lien HSL. L'automate du composant PCI-DDC positionne ce bit, au moment de la construction du paquet.

L'utilisation de cette file d'attente par l'automate est simple :

- Dans un premier temps, l'automate écrit l'en-tête du paquet, défini au paragraphe 4-2-2.

emplacements libres. Ainsi l'automate peut écrire les caractères de fin de paquet sans tester le débordement. Il fournit de même un drapeau qui indique si le nombre d'emplacements libres dans cette file d'attente dépasse un certain seuil, pour autoriser la procédure d'envoi de paquets.

Nous étudions les répercussions de la profondeur des files d'attente sur les performances au chapitre 6.

4-3-3 Gestion du flux.

Comme le routeur RCube, Le composant PCI-DDC gère le contrôle de flux en respectant le standard IEEE 1355. Ses files d'attente peuvent contenir plus de deux « flits » de 32 caractères chacun. En effet, pour masquer la séquentialisation des réceptions et des émissions due au fait que le bus PCI ne permet d'effectuer des transferts qu'alternativement dans les deux directions, ces files d'attente présentent une capacité de mémorisation importante.

RCube ne peut compter plus de 2 « flits » de crédit d'émission, soit 64 caractères. C'est-à-dire qu'il ne peut pas tirer parti de la grande taille de la file d'attente de réception de PCI-DDC. Ceci est principalement dû à l'absence de recommandations dans la norme IEEE 1355, qui ne spécifie pas la taille de ces compteurs. PCI-DDC va donc se comporter exactement, vu de l'extérieur, comme si sa file d'attente de réception ne comportait pas plus de deux flits, bien que sa capacité soit bien supérieure.

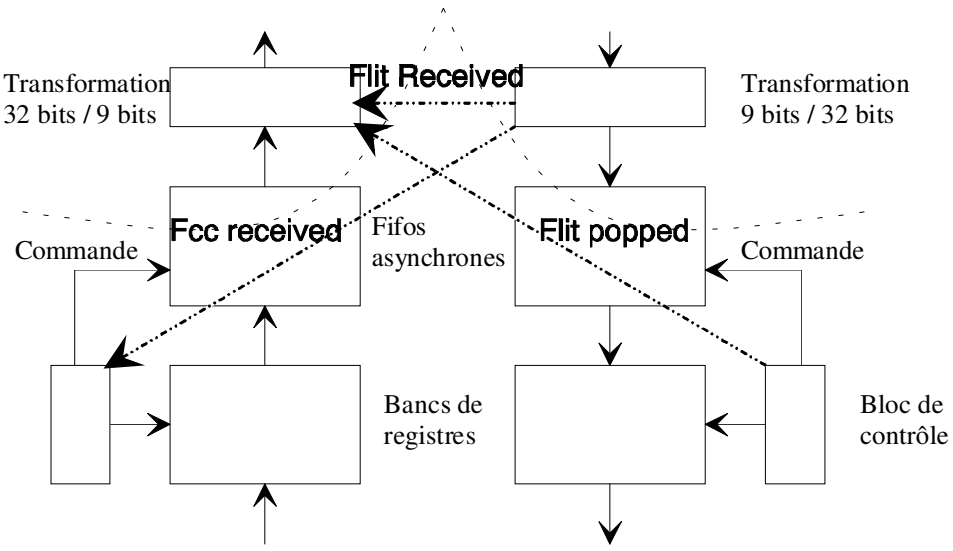

Figure 4-m : gestion du contrôle de flux.

- Pour gérer le contrôle du flux entrant dans le composant, PCI-DDC tient un compteur à trois positions dans le bloc de transformation 32 bits vers 8 bits. Ce compteur est incrémenté à chaque réception d'un flit, réception qui lui est signalée par « Flit Received » et décrémenté à chaque extraction d'un flit complet de la file d'attente synchrone d'entrée, signalée par « Flit Popped ». Ce compteur indique donc le nombre de « flits » présents dans les files d'attente asynchrones de réception. A chaque réception de « Flit Received », un caractère de contrôle de flux est envoyé, si le compteur indique la présence d'au moins un « flit » de libre dans ces files d'attente. Ainsi, PCI-DDC semble n'avoir qu'une file d'attente de deux flits de capacité, alors que sa capacité est plus importante.
- En émission, PCI-DDC gère un compteur de crédit. Le signal « FCC Received » est transmis au bloc qui pilote les écritures depuis la file d'attente synchrone vers la file

La figure 4-k présente les relations entre ces trois horloges. Deux files d'attente asynchrones de largeur 4 octets permettent de transmettre et de recevoir des données. Ces files d'attente sont semblables à celles utilisées dans le routeur RCube, avec une largeur différente. Elles permettent de régler les problèmes électriques dûs à la présence de plusieurs horloges dans le transfert des données.

Dans tout le composant, les données manipulées sont des mots de 32 bits. Le paragraphe 4-3-4 détaille la mise en forme des mots de 8 bits en mots de 32 bits, ainsi que la gestion des problèmes de non-alignement de données sur des frontières de mots.

Les mécanismes de signalisation fournis par les files d'attente asynchrones sont malheureusement très limités : seuls les états « totalement plein » et « totalement vide » sont détectés lors de l'écriture ou de la lecture du dernier mot. Ces informations sont insuffisantes pour nos besoins :

- Dans le cas de l'émission de paquets, la condition « file d'attente pleine » doit conduire à une fin de paquet, pour permettre au composant d'effectuer des réceptions de paquets. Comme cette file d'attente est pleine, il est difficile d'y introduire les derniers octets du paquet, c'est-à-dire le code détecteur d'erreur et le caractère de fin de paquet. L'automate doit donc attendre que celle-ci se vide pour y introduire les caractères manquants, interdisant donc des réceptions concurrentes.
- Par ailleurs, il est inutile de transmettre des paquets trop petits, car le surcoût dû au nombre de caractères dans l'en-tête et la queue du paquet devient trop importante. Avant donc de commencer l'envoi d'un paquet, il est intéressant d'exiger au moins un certain nombre de mots disponibles dans la file d'attente d'émission. La file d'attente asynchrone ne fournit pas cette information.

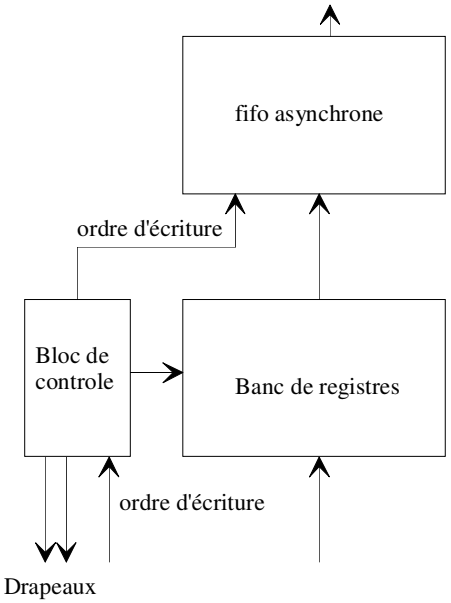

Figure 4-l : coopération de la file d'attente asynchrone et du banc de registres à l'émission.

Notre politique consiste, pour éviter ces phénomènes, à accoupler à cette file d'attente asynchrone une file d'attente synchrone (en fait un banc de registres). Un bloc de contrôle gère les pointeurs de la file d'attente synchrone et connaît donc à chaque cycle le nombre de mots présents. Il fournit à l'automate un drapeau « Presque Plein » lorsqu'il n'y a plus que trois

Le composant d'interface PCI-DDC.

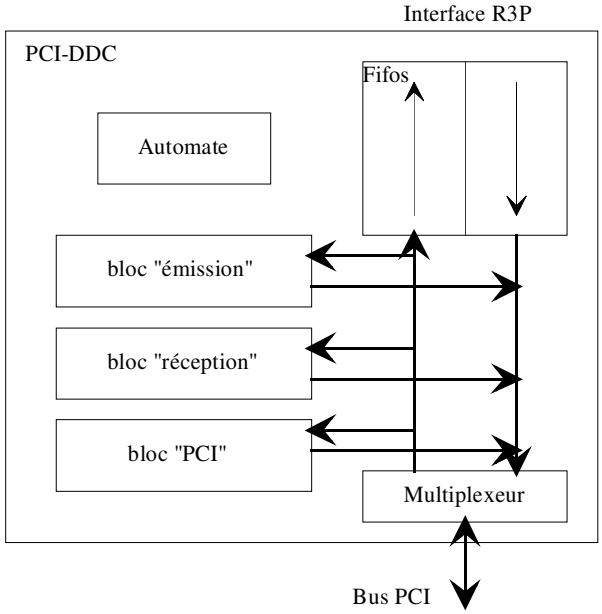

Figure 4-j : schéma bloc de PCI-DDC.

4-3-2 Interface vers le réseau HSL.

Le composant PCI-DDC fonctionne avec trois horloges différentes.

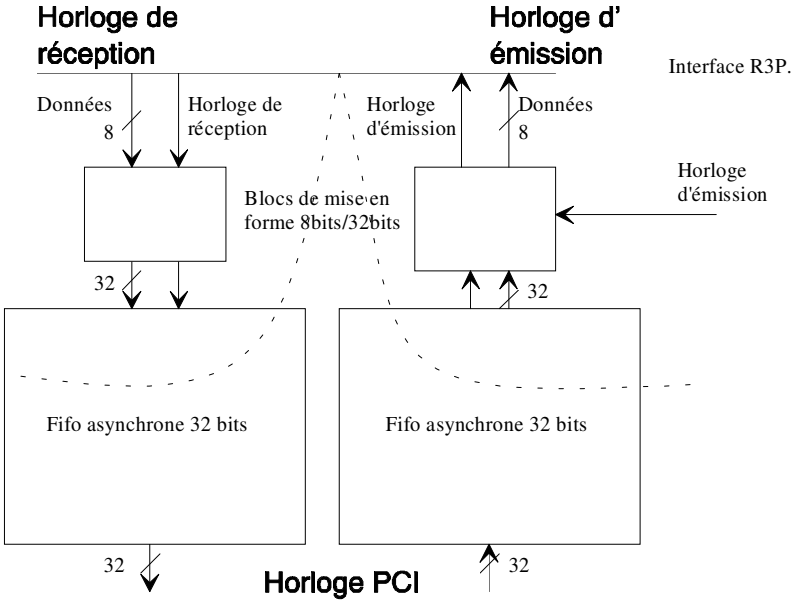

Figure 4-k : les trois horloges de PCI-DDC.

- La première de ces horloges est celle du bus PCI, elle cadence la plus grande partie des opérateurs. PCI-DDC écrit et lit des données dans les deux files d'attente reliées au réseau HSL à cette fréquence.
- L'émission des données vers le réseau HSL utilise une horloge à 80 Mhz fournie par un oscillateur externe qui permet l'adaptation au débit du lien HSL.
- Une troisième horloge est extraite du lien HSL en réception.

paquets ainsi formés sont marqués d'un identificateur de message spécifique, pour permettre au composant PCI-DDC B de les distinguer.

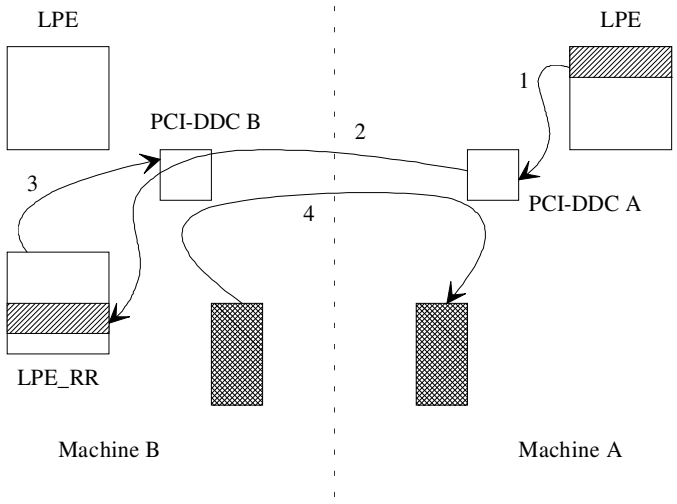

Figure 4-i : Lecture distante.

- 3) Le composant PCI-DDC B gère une seconde table LPE_RR, spécifique pour les lectures distantes. A la réception du message contenant le descripteur, il place celui-ci dans sa liste LPE_RR. Comme les adresses « locale » et « distante » ont été inversées dans le descripteur à l'émission, le descripteur présente un format totalement identique à ceux de la LPE de B.
- 4) Le composant PCI-DDC B effectue ensuite une écriture à distance comme présentée plus haut dans ce chapitre, si ce n'est que le descripteur a été reçu du réseau, et non construit par le processeur local.

Dans la version actuelle de PCI-DDC, ce mécanisme peut être émulé par les logiciels de bas niveaux. Fondamentalement, l'écriture à distance est la plus petite opération de transfert de données possible. D'autres, comme la lecture à distance, peuvent être construites en l'utilisant.

4-3 Architecture Interne.

4-3-1 Présentation.

La complexité du composant PCI-DDC ne se situe pas sur les mêmes plans que celle du routeur RCube.

Dans ce dernier, la concurrence entre les automates, la minimisation de la latence, ainsi que la fréquence élevée de l'horloge (80 Mhz) étaient les problèmes principaux. Le circuit d'interface PCI-DDC fonctionne en grande partie à la fréquence du bus PCI, c'est-à-dire 33 Mhz. La difficulté réside ici dans le respect de la norme PCI, la construction de paquets de taille variable, et donc dans la complexité de l'automate de contrôle.

Sur la figure 4-j, nous présentons un schéma bloc de PCI-DDC.

Un automate principal contrôle le composant. Il dispose de trois blocs contenant les opérateurs associés aux différentes tâches qu'il accomplit : l'émission des paquets, la réception des paquets et la gestion du bus PCI. Deux files d'attente, l'une en émission, l'autre en réception, permettent la communication avec le réseau HSL.

4-2-4 La barrière de synchronisation.

Une autre opération souvent rencontrée dans les programmes parallèles est la barrière de synchronisation. Plusieurs processus doivent se synchroniser sur la terminaison d'un sous-programme, par exemple.

Notre interface propose une primitive qui permet d'implémenter simplement des barrières de synchronisation. On suppose qu'un processus attend un message de *p* autres processus, pour pouvoir continuer un traitement. Ce processus alloue une entrée IM locale, dont le champ NPA, au lieu d'être remis à zéro comme pour les réceptions normales, contient la valeur *p*.

Les *p* machines qui doivent se signaler, émettent alors un message, composé d'un unique paquet (car de taille égale à un mot), en écrivant une entrée dans la LPE en positionnant le drapeau CM à 1. Un paquet est alors envoyé, de même format que les autres si ce n'est que le champ NPN est à zéro, bien que ce paquet soit le dernier (parce que le seul) du message.

Ainsi, la réception de ces *p* paquets, émis par les *p* machines qui devaient se signaler, se manifestera sur la machine réceptrice par l'égalité des champs NPR et NPA dans la ligne IM.

Ce mécanisme, dont la complexité matérielle se résume à ne pas envoyer le nombre de paquets nécessaires à l'envoi du message dans le champ NPN, permet de générer efficacement des barrières de synchronisation matérielles. Dans l'exemple d'interface logicielle que nous présentons dans le chapitre 5, cette primitive n'est pas utilisée.

4-2-5 Lectures distantes.

Le composant PCI-DDC ne propose qu'une seule primitive : l'écriture à distance. Une autre primitive classiquement utilisée, la lecture à distance, peut être facilement implémentée grâce à notre dispositif.

La lecture à distance peut être vue comme deux écritures successives :

- Dans un premier temps, un processeur A désirant lire la mémoire d'un processeur B effectue l'envoi à celui-ci d'un message, contenant le descripteur de la zone à lire dans B, ainsi que le descripteur de la zone de mémoire dans A où placer les données.
- Dans un second temps, la machine B, à la réception de ce premier message, effectue l'envoi des données, comme une écriture à distance, vers la zone mémoire de A désignée dans le premier message.

Dans ce cas, un traitement logiciel est effectué par le gestionnaire d'interruption de la machine B. C'est en effet celui-ci qui analyse le premier message, et place dans la LPE les descripteurs permettant l'envoi effectif des données.

Si la lecture distante apparaît comme une primitive largement utilisée, il est possible de rajouter un support matériel dans le composant PCI-DDC permettant d'éviter le traitement logiciel décrit ci-dessus. Ce mécanisme, qui sera peut-être intégré dans une future version de PCI-DDC, est présenté succinctement sur la figure 4-i.

- 1) La machine A désire lire une portion de la mémoire de la machine B. Elle construit un descripteur spécifique, marqué d'un drapeau spécial, et le place dans sa propre LPE. Ce descripteur est identique aux descripteurs présentés dans le paragraphe 4-2-2, si ce n'est que les adresses « locale » et « distante » sont inversées.
- 2) Le composant PCI-DDC A remarque le drapeau spécial dans le descripteur. Au lieu d'envoyer effectivement des zones de mémoire, il envoie le descripteur lui-même. Les

4-2-3 Méthode de réception.

La tâche du composant PCI-DDC, du côté du récepteur, est plus simple. Il lui suffit d'écrire les données reçues à l'adresse indiquée dans l'en-tête du paquet. Cependant, il doit pouvoir générer une interruption lorsque la totalité des paquets associés à un message a été reçue, de manière à éventuellement réactiver le processus en attente du message.

Pour cela, il maintient une table, la LMR pour Liste des Messages en Réception, elle aussi contenue dans la mémoire principale. Elle possède le format décrit sur la figure 4-g :

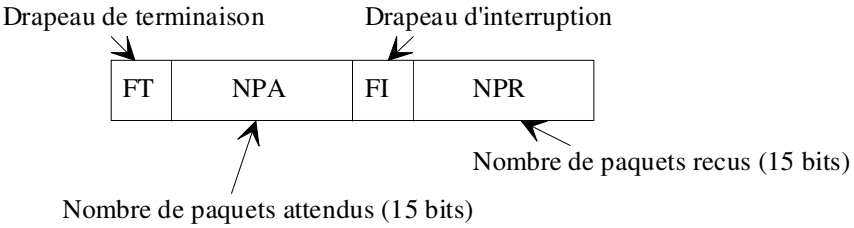

Figure 4-g : format de la LMR.

Une telle structure est associée à chaque message en cours de réception. Cette ligne est tout d'abord initialisée à zéro par le système récepteur du message. Tous les champs prennent la valeur zéro, sauf l'indicateur d'interruption qui signale que le composant doit éventuellement générer une interruption à la fin de la réception de ce message.

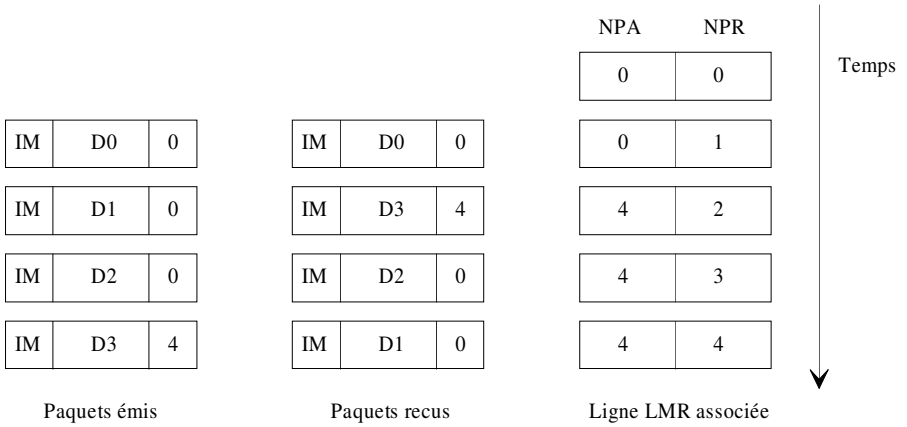

Figure 4-h : exemple de réception.

A chaque réception d'un paquet, le champ NPR associé à la ligne du IM du paquet est incrémenté par le composant PCI-DDC, et le champ NPA prend la valeur NPN si celle-ci est non nulle.

Lorsque les deux champs NPR et NPA sont égaux, c'est que la totalité des paquets associés à ce message a été reçue, même si leur ordre d'arrivée a été perturbé par le réseau HSL.

On peut voir sur la figure 4-h un exemple de détection de fin de message avec déséquentialisation des paquets.

de la figure 4-f . On constate que le format de paquet PCI-DDC est plus complexe que le format de paquet utilisé par le routeur (le format de paquet pour RCube ne définit que les champs DEST et EP).

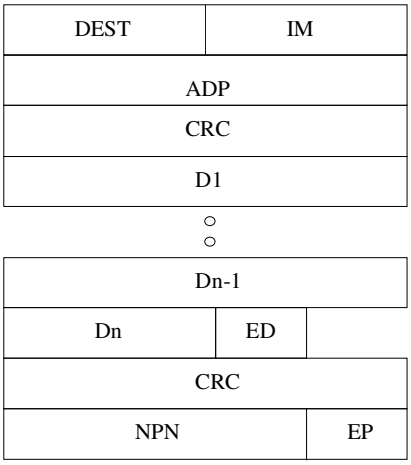

Figure 4-f : format des paquets.

- DEST, IM, ADP ont la même signification que ceux du descripteur LPE.
- CRC : code détecteur d'erreur pour vérifier la validité de l'adresse, du destinataire et de l'identificateur de message.
- D1 - Dn : données à transmettre, pas forcément cadrées sur des mots de 32 bits.
- ED : caractère spécifique IEEE 1355, 9 bits, « fin de donnée ».
- NPN : nombre de paquets du message. Tous les paquets ont ce champ à zéro, sauf le dernier paquet émis d'un message, qui contient le nombre de paquets émis pour ce message.
- EP : il s'agit du caractère de fin de paquet IEEE 1355.

Le contrôleur PCI-DDC tente de construire les paquets les plus gros possibles. Il effectue un accès « rafale » sur le bus PCI et construit le paquet au fur et à mesure qu'il récupère les données. La norme PCI prévoit cependant qu'un composant ne peut garder le contrôle du bus plus d'un certain temps. C'est-à-dire que, parfois, le contrôleur PCI-DDC doit relâcher le bus avant que la totalité des données de la page ait été transmise. Deux solutions sont possibles :

- Le composant PCI-DDC peut ne pas terminer le paquet, et laisser ainsi la liaison « ouverte », en insérant des caractères IDLE. Parallèlement, il tente de reprendre le bus PCI. Avec le routage wormhole, cette technique est extrêmement mauvaise, car les liens sont utilisés à transmettre des caractères inutiles.
- La solution retenue consiste à terminer le paquet afin de libérer le réseau. Les liens HSL sont donc disponibles pour router d'autres paquets. Le composant PCI-DDC redemande le bus pour reconstruire un autre paquet avec la suite des données. Ce nouveau paquet portera dans son en-tête l'adresse mémoire où il faut le placer, c'est à dire l'adresse du début de la zone de réception de la page, augmentée du nombre d'octets déjà transmis. Cette technique implique, entre autre, qu'aucun des deux intervenants ne connaît a priori le nombre de paquets nécessaire au transfert d'un message.

Ainsi, l'envoi d'un message par le logiciel génère un certain nombre de paquets, portant tous le même IM, identificateur de message, et dont le dernier émis contient dans son champ NPN le nombre de paquets qu'il a été nécessaire d'émettre pour envoyer la totalité du message.

Les processus s'échangent des messages. Ils n'ont pas connaissance de la segmentation de la mémoire en pages. Or notre système de « Direct Deposit » impose la connaissance des adresses physiques. Donc, avant d'effectuer le transfert des données, les processus émetteur et récepteur vont verrouiller les zones d'émission et de réception, c'est-à-dire s'assurer qu'elles ne seront pas déplacées par le gestionnaire de la mémoire virtuelle pendant la durée du transfert. La décomposition du message en pages est assurée par le logiciel, non seulement pour des raisons de simplification du matériel, mais aussi parce que cette décomposition dépend du type du processeur.

Une entrée dans la LPE possède une taille de quatre mots de 32 bits, dont le format est donné sur la figure 4-e[36] .

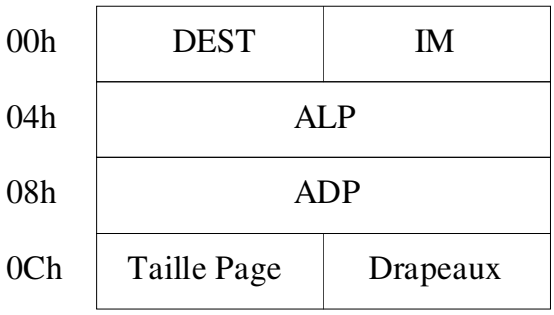

figure 4-e : format de la LPE.

- « DEST » est un champ de 16 bits. Il constitue l'en-tête, au sens IEEE 1355 des paquets, c'est-à-dire l'adresse HSL du destinataire.
- « IM » est un identificateur de messages. Si un message est composé de plusieurs pages, toutes celles-ci possèdent le même IM. Tous les paquets qui seront émis transportant des parties d'une de ces pages porteront aussi ce IM.
- « ALP » est l'adresse physique locale du début de la page à transmettre. On voit ici que, dans sa version actuelle, le composant PCI-DDC s'adresse aux systèmes à adressage 32 bits. Cette adresse n'est pas forcément alignée sur une frontière de mot.
- « ADP » est l'adresse du début de la page dans la mémoire du récepteur.
- « Taille de la page » est le nombre d'octets à émettre.
- « Drapeaux »: ces indicateurs permettent une gestion fine du protocole.
    - IOS : une interruption doit être générée sur le système local après l'envoi complet de cette page.
    - LMP : cette page est la dernière associée au message IM.
    - CM : ce message est d'un type spécial, que nous définissons au paragraphe 4-2-4.

Pour envoyer un message, le système écrit un certain nombre de descripteurs dans la table LPE. Puis il informe le contrôleur réseau, via une écriture dans un registre du composant PCI-DDC, du nombre de descripteurs ajoutés. Ensuite le composant effectue un accès en mémoire pour lire le premier descripteur. Il effectue alors l'envoi des données, avec le format de paquet

[36] Les formats de données peuvent être amenés à changer pour simplifier la réalisation du composant, les informations présentes restant, en tous cas, les mêmes.

sur le bus PCI. Le système rajoute à cette liste un ou plusieurs descripteurs, par des accès mémoire bien plus rapides que des accès en entrée-sortie. Ensuite, il effectue une unique écriture dans l'espace de configuration du composant PCI-DDC pour lui communiquer le nombre d'entrées ajoutées dans cette liste. Ainsi le nombre d'accès I / O, fort pénalisant, est minimisé. Le composant PCI-DDC ne doit donc contenir que des pointeurs sur cette liste, pour pouvoir aller chercher les descripteurs des messages à émettre, et connaître en permanence le nombre d'entrées à traiter.

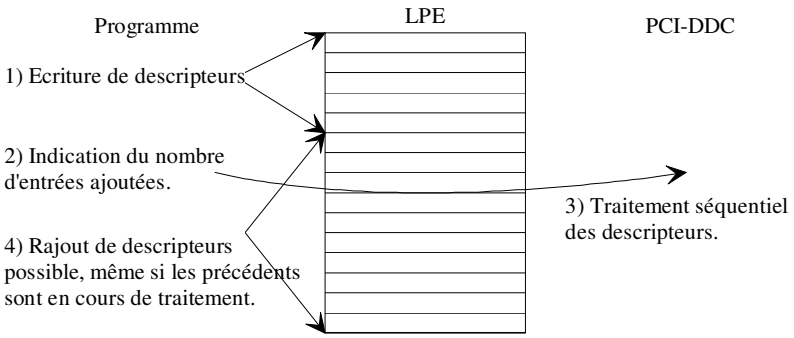

Figure 4-d : gestion de la Liste des Pages en Emission.

Cependant, un inconvénient de ce système est qu'il nécessite obligatoirement un appel système de la part du processus émetteur pour émettre un message, puisque la LPE est une ressource centrale du système d'exploitation, partagée par tous les processus. De nombreux projets[35] ont essayé d'éviter ces appels systèmes, pour minimiser l'« Overhead », en donnant l'accès à certaines ressources de leur contrôleur réseau aux processus en mode utilisateur, par exemple en rendant accessibles les registres de contrôle dans l'espace mémoire utilisateur.

Notre méthode présente l'avantage de minimiser la complexité du matériel, d'éviter la présence de mémoire sur le périphérique, d'éviter des accès d'entrée-sortie inutiles. D'autre part la LPE réside dans la mémoire principale, et peut donc être de grande taille.

De même, la structure de données associée aux messages en réception, la LMR (liste des messages en réception décrite en 4-2-3), qui est écrite par PCI-DDC et lue par le logiciel, réside en mémoire principale.

4-2-2 Méthode d'émission.

Tout d'abord nous introduisons les concepts de messages, pages et paquets.

- Le message est l'unité d'échange entre deux processus. Il est de taille quelconque. Il se peut que le message soit d'une taille supérieure à celle des pages de mémoire virtuelle du système, et donc segmenté en mémoire physique.
- Une page est une zone de mémoire physique contiguë d'un message. Le découpage d'un message en pages est assuré par le logiciel. Une page ici ne correspond pas forcément aux pages telles qu'elles sont définies par le mécanisme de mémoire virtuelle.
- Un paquet est une unité d'échange dans le réseau, c'est-à-dire les paquets tels que nous les avons définis dans la section 3-1-2. Une page peut être émise en plusieurs paquets, comme nous l'expliquons dans la suite de ce paragraphe. Le découpage d'une page en paquets est assuré par le matériel.

[35] [SWAN95]

en rajoutant des synchronisations entre l'émetteur et le récepteur. Enfin, dernier exemple, il alloue une portion de la mémoire physique pour les transferts, comme c'est le cas dans Myrinet, et accepte d'effectuer des recopies.

Dans le cas de machines parallèles effectuant des traitements particuliers, comme c'est le cas de la machine du CERN, ou de Thomson ASM, ce verrouillage n'est pas un handicap. Ici, les machines sont dimensionnées pour effectuer un certain traitement. L'accélération des transferts et la diminution de la charge du processeur qui n'effectue pas de recopies inutiles sont donc directement « payées » par le verrouillage de zones de mémoire, et l'éventuelle augmentation de la mémoire du système au profit des performances.

Conclusion.

Le système construit sur notre réseau d'interconnexion, en fonction de ses besoins, peut utiliser de manière plus ou moins directe le mécanisme de recopie directe en mémoire.

Dans les cas où le système ne peut pas permettre de rendez-vous préalable ou bien ne peut pas verrouiller de larges zones de mémoires (la mémoire est une ressource critique), une solution similaire à celle de Myrinet doit être mise en œuvre : une portion de mémoire est allouée, à la manière d'un tampon de réception, pour effectuer les transferts. On perd alors le gain proposé de s'affranchir des recopies, tout en conservant les deux autres avantages : la reconstruction automatique des paquets en messages et la diminution du nombre d'interruptions logicielles.

Dans les cas où cela est possible, ce mécanisme permet de diminuer la latence des transferts et « l'Overhead » généré par le transfert des messages, en évitant des recopies inutiles effectuées par le processeur.

4-2 Spécification.

4-2-1 L'interface au bus PCI.

Un point important est la manière dont le composant PCI-DDC, qui implémente notre protocole, s'interface au bus PCI et au système.

La totalité des transferts de données, c'est-à-dire la prise des données depuis l'espace de mémoire du processus émetteur et leur écriture dans l'espace de mémoire du processus récepteur, est effectuée par le composant d'interface en mode maître. Le composant PCI-DDC effectue des accès en rafale sur le bus PCI pour lire ou écrire les données.

Pour qu'un processus signale qu'il a une émission à effectuer, il doit préparer, comme nous le verrons en détail dans le paragraphe suivant, une structure de données qui contienne les descripteurs de la zone mémoire à émettre, ainsi que la description de son emplacement de réception dans la mémoire du destinataire.

Traditionnellement, le programme écrit ce descripteur dans une zone d'entrée-sortie du périphérique. L'« Overhead » est accru par les accès I / O qui sont effectués. D'autre part, si l'on veut que le logiciel programme plusieurs émissions en une seule opération, le contrôleur réseau doit contenir une mémoire pour ranger les différents descripteurs, ce qui est coûteux, ou encore pire, interdire l'ajout d'un descripteur si le précédent est encore en traitement.

Nous employons une autre méthode, consistant à ranger les descripteurs de messages à émettre en mémoire principale. C'est la LPE, Liste des Pages en Emission. Celle-ci est écrite par le système d'exploitation, et lue par le composant PCI-DDC à l'aide d'un accès en mode rafale

Dans le cas où un message est segmenté en paquets lors de son émission, le temps de reconstruction du message en réception devient quasiment nul. Le contrôleur réseau en réception analyse l'en-tête, et range directement les paquets à leur place dans la mémoire. Ceci règle aussi le problème de déséquentialisation des paquets dans le réseau d'interconnexion. Il faut cependant proposer un mécanisme pour s'assurer que tous les paquets associés à un message ont été reçus. Notre protocole ne génère alors qu'une seule interruption pour la réception d'un message complet, et non une par paquet comme dans les protocoles classiques.

Ces deux concepts, « Direct Deposit », et « Stateless Receiver » forment le cœur de notre système de communication : « Direct Deposit Stateless Receiver Protocol », ou DDSLRP.

Le composant PCI-DDC ne présente donc qu'une seule primitive : l'écriture à distance, où une machine peut transférer des données qui résident dans sa mémoire locale vers la mémoire distante d'une autre machine. Le chapitre 5 présente un exemple de mécanisme logiciel qui permet d'effectuer des opérations usuelles d'émission et de réception de messages dans un modèle de programmation « send/receive » en n'utilisant que cette primitive.

4-1-5 Conséquences.

Ce protocole de communication, qui transporte directement des données depuis une zone de mémoire où l'adresse physique est connue vers une autre zone à adresse physique connue implique certaines conséquences.

Rendez-vous préalables.

Il faut, pour effectuer le transfert, que l'émetteur ait connaissance des adresses physiques dans la mémoire du destinataire. Ceci suppose un rendez-vous préalable au transfert effectif des données entre l'émetteur et le récepteur du message. Durant ce rendez-vous le récepteur a transmis à l'émetteur l'emplacement de la zone de réception des données. Ce rendez-vous préalable supprime les restrictions de taille des mémoires tampon de réception dans les protocoles classiques. Un autre avantage de ce rendez-vous est la disparition des acquittements de réception.

En fait, on échange les acquittements postérieurs à la communication par des acquittements préalables. La mise en place de solutions dans les protocoles utilisant notre réseau d'interconnexion, (comme le fenêtrage dans TCP par exemple), permet de masquer le temps d'initialisation de la communication, en parallélisant le transfert des messages préalables de signalisation avec le transfert effectif des données.

Verrouillage des zones d'émission et de réception.

Les processus qui sont en attente de messages, ou qui en émettent, sont obligés de verrouiller en mémoire les zones de réception ou d'émission. Les pages correspondantes de la mémoire physique ne sont donc pas utilisables pour d'autres processus.

Dans le cas de machines parallèles génériques, utilisant des systèmes d'exploitation standards, et supportant simultanément avec l'exécution de traitements parallèles des charges séquentielles, ce verrouillage se fait au détriment d'autres applications.

Le système d'exploitation doit alors mettre en œuvre des politiques plus fines de gestion de ce mécanisme. Il limite, par utilisateur, les portions de mémoires qui peuvent être verrouillées, ou bien, autre exemple, il fragmente les transferts de large taille en transferts de plus petite taille, permettant un verrouillage successif de différentes sous-zones d'émission et de réception, mais

Plus une machine reçoit de paquets, plus elle en détruit, ce qui va nécessiter leur réémission, c'est-à-dire encore plus de paquets à recevoir et une occupation accrue de la bande passante du réseau.

4-1-3 « Direct Deposit Protocol ».

La première réponse que nous apportons est le « Direct Deposit Protocol ». Comme on le voit sur la figure 4-c.

> ***Le composant d'interface peut transporter directement les données depuis l'espace mémoire du processus émetteur vers l'espace mémoire du processus récepteur, sans autre recopie.***

Ce transfert est assuré par le composant d'interface PCI-DDC, qui va directement chercher les données dans la mémoire du processus émetteur, les émet sans recopie mémoire, et les dépose dans la mémoire du processus récepteur, là aussi sans recopie mémoire. Ce mécanisme, que nous exposons en détail dans la suite du chapitre, a un faible coût matériel car il n'implique pas la présence de mémoire spécifique sur la carte d'interface. Il peut réduire la latence des transferts d'un facteur 5, car il évite 4 recopies qui s'effectuent avec un débit comparable à celui de la liaison HSL. Il réduit considérablement l'« Overhead », dans la mesure où le processeur principal n'a pas à effectuer de mouvements de données.

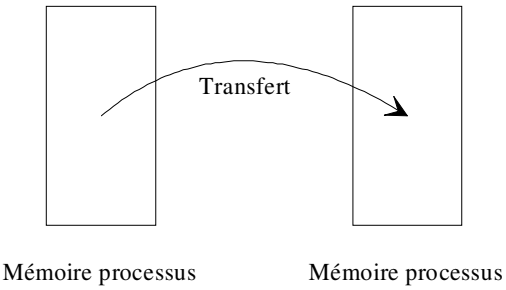

Figure 4-c : Direct Deposit Protocol.

4-1-4 « Stateless Receiver Protocol ».

Dans le protocole IP, les fragments de datagrammes IP contiennent une adresse relative dans la mémoire du destinataire. Mark Swanson, de l'université de l'Utah, reprend ce concept pour les réseau d'interconnexion de machines parallèles, et le nomme « Sender Based Protocol »[33]. Chaque paquet qui transite sur le réseau contient sa position dans une mémoire tampon de réception réseau. Ceci permet de diminuer les temps de reconstruction des messages en paquets. Nous généralisons ici ce principe.

> ***Chaque paquet contient dans son en-tête l'adresse dans la mémoire physique du récepteur où il faut le placer.***

Nous appelons ce mécanisme « Stateless Receiver Protocol », car le récepteur des messages n'effectue presque aucun traitement sur les paquets, il se contente de les mettre en mémoire à l'adresse indiquée dans l'en-tête. Ceci peut aussi être vu comme un cas particulier des « Active Messages » de Eicken[34].

[33] [SWAN94SBP] et [SWAN93PPE]
[34] [EICKEN]

4-1-2 Les interfaces classiques.

Les interfaces entre le réseau et la cellule processeur présentent parfois certains inconvénients. Nous proposons, pour notre composant d'interface, un support matériel permettant de mettre en œuvre des techniques plus efficaces de transferts de données.

- Dans la plupart des cas, comme on peut le voir sur la figure 4-b, de nombreuses recopies de zone mémoire sont effectuées dans les transferts de données. C'est en particulier le cas pour les protocoles implémentés sur les réseaux locaux. En effet les données à transférer sont d'abord recopiées depuis la mémoire du processus émetteur vers une mémoire tampon du système pour effectuer la mise en forme des paquets, puis parfois dans une mémoire tampon présente sur le contrôleur réseau. Du coté du récepteur, elles sont là aussi recopiées depuis la mémoire du contrôleur vers le noyau, puis enfin du noyau vers l'espace mémoire du processus récepteur. Souvent, les recopies entre la mémoire spécifique du contrôleur réseau et le noyau sont effectuées par le matériel, déchargeant ainsi le processeur. Le temps de recopie d'une zone mémoire vers une autre, compte tenu de la bande passante limitée des bus des stations de travail (100 MOctets / sec), est du même ordre de grandeur que le transfert à 1 Gbit / sec des données. Cette technique n'est plus envisageable avec des débits tels que les nôtres, puisque chacune des recopies a une latence comparable à celle du réseau.

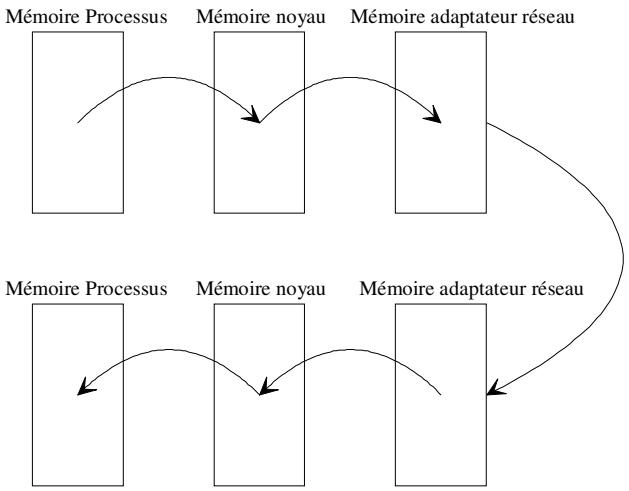

Figure 4-b : recopie dans les protocoles classiques.

- La reconstruction des messages à partir de paquets est en général assurée par le logiciel. Chaque paquet reçu génère une interruption, dont le traitement dans les systèmes d'exploitation est souvent très coûteux, suivie d'opérations logicielles pour reconstruire le message à partir des paquets.
- La taille de la mémoire tampon sur la carte réseau et dans le noyau est nécessairement limitée. Les messages de taille supérieure sont donc tronçonnés. Parfois ces mémoires tampon sont pleines. Les paquets dont la réception entraîne le débordement sont donc détruits. Ce mécanisme impose la mise en place d'un coûteux protocole d'acquittement et de réémission des données, dont le traitement est parfois assuré par logiciel.

Ces deux derniers points se combinent dans un cercle vicieux. Il faut du temps (traitement de l'interruption et opérations logicielles) pour extraire un paquet de l'interface réseau. Pendant ce temps, les autres paquets en réception sont détruits car les mémoires tampon sont pleines.

Modes de transfert.

On peut distinguer trois types de transferts mettant en jeu un périphérique, suivant le composant qui en a pris l'initiative.

- Le microprocesseur effectue des accès d'entrée-sortie sur un périphérique PCI. C'est typiquement le cas lorsqu'une carte vidéo est présente sur le bus PCI : le microprocesseur accède à un certain nombre de registres du composant vidéo par des instructions spécifiques d'entrées-sorties (IN et OUT sur les processeurs INTEL). Le bridge PCI transforme ces accès en requêtes PCI. Ce mode est spécialement peu performant. Il permet essentiellement de garder la compatibilité des logiciels pour les versions PCI de cartes existant pour d'autres bus.
- Une partie de l'espace d'adressage du système, c'est-à-dire une partie de la mémoire physique, peut être déclarée comme étant présente sur un périphérique PCI. Le bridge, alors, va transformer les instructions de lecture et d'écriture du processeur vers la mémoire en requêtes PCI. Ce mode est légèrement plus performant que le précédent, dans la mesure ou le bridge essaie de construire des accès « rafale ». Evidemment, celui-ci ne peut savoir a priori à quelle zone exacte veut accéder le logiciel, donc les rafales sont de petite taille.
- Un troisième mode, de loin le plus intéressant, est celui où un périphérique accède directement à la mémoire principale en mode maître. Il peut donc initier un transfert de grande taille, sur un bloc par exemple, dans la mesure où, contrairement au bridge, il connaît a priori la taille du transfert à effectuer. Le bus PCI permet donc aux périphériques qui y sont connectés de programmer eux-mêmes des accès depuis et vers la mémoire, les contrôleurs de DMA supplémentaires ne sont donc plus nécessaires. D'autre part, le bridge est en général intégré au contrôleur de cache L2. C'est donc ce dernier qui assure la cohérence de la mémoire, y compris lors d'accès concurrents du processeur et des périphériques PCI. Donc les périphériques bénéficient eux aussi du cache L2 (en général, en lecture uniquement), ce qui permet d'accélérer les transferts. Ce mode est de loin le plus performant. C'est celui que nous allons utiliser pour notre interface réseau.

Un quatrième type de transfert est présent, bien que marginal. Chaque périphérique PCI doit présenter un certain nombre de registres de configuration, qui peuvent être écrits ou lus par le microprocesseur. Ceux-ci permettent, entre autres, d'identifier les composants présents ou de leur allouer au démarrage du système des zones de mémoire. Il est obligatoire d'implémenter ces registres, dont un certain nombre sont cependant spécifiques à chaque périphérique.

Un bus évolutif.

On voit donc clairement que le « bridge CPU-PCI » est un composant central du système. Les différentes implémentations effectuées par les constructeurs sont de plus en plus performantes, et incluent de plus en plus d'optimisation (construction de burst, lectures anticipées, écritures postées etc.). Il existe des bridges PCI pour chaque grande famille de processeurs du marché, INTEL, DEC, Motorola. Le choix du bus PCI assure la pérennité de notre système. Un bridge spécifique est en général disponible en même temps qu'un nouveau microprocesseur. D'autre part, le bus est défini jusque 66 Mhz et 64 bits, ce qui assure un transfert maximum de 528 MOctets / sec. Les versions actuelles sont à 33 Mhz sur 32 bits. Certains constructeurs annoncent déjà les versions 66 Mhz 32 bits.

Le bus PCI est donc un choix très intéressant. Il nous assure de bénéficier rapidement des évolutions des processeurs et ses performances sont bien adaptées à la bande passante du réseau d'interconnexion.

Chapitre 4 : Le composant d'interface PCI-DDC.

Après avoir défini les fonctionnalités du routeur RCube, nous allons maintenant spécifier celles du composant d'interface entre le réseau d'interconnexion et la cellule processeur.

4-1 Motivation

Nous avons vu dans le chapitre 1 que l'interface entre la cellule processeur et le réseau d'interconnexion doit se faire par l'intermédiaire d'un bus standard. Le bus PCI, standard de fait de l'industrie, est celui que nous avons retenu.

4-1-1 Le bus PCI.

Le bus PCI est un bus 32 ou 64 bits, à adresses et données multiplexées[32]. Il peut opérer en 3,3V ou 5V. C'est un bus multi-maître, c'est-à-dire que plusieurs périphériques présents sur le bus peuvent initier des transferts. Il présente des modes de transferts de données par rafale, ou « burst », dans lesquels une seule adresse est échangée, suivie de données consécutives en mémoire. Sur la figure 4-a, nous présentons l'architecture d'une machine à bus PCI.

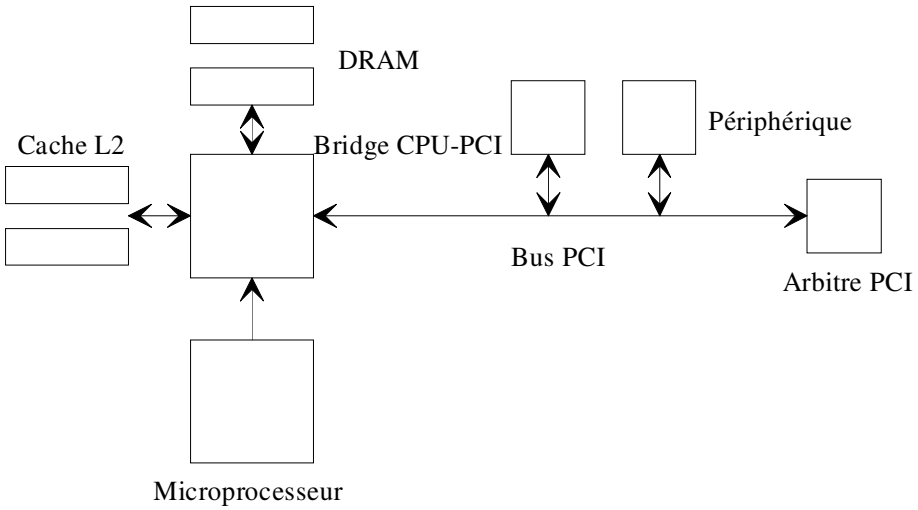

figure 4-a : architecture d'un monoprocesseur à bus PCI.

Nous pouvons identifier deux éléments clés d'un système à bus PCI.

- Tout d'abord l'arbitre. Chacun des périphériques maîtres peut demander à utiliser le bus. Pour cela, ils effectuent des requêtes. Celles-ci sont gérées par l'arbitre, qui assure une répartition équitable du bus.
- Le « bridge CPU-PCI ». Ce composant gère le cache de deuxième niveau et la mémoire principale du système. Il permet aussi au processeur d'effectuer des accès sur le bus PCI et aux périphériques d'accéder à la mémoire principale, en bénéficiant eux aussi du cache de deuxième niveau. Ce composant est spécifique pour chaque type de microprocesseur.

[32] [PCI]

interlocuteur IEEE 1355. Une fois que la calibration de leur propre désérialiseur est effectuée, ils effectuent un échange de caractères de contrôle spécifiques, « START_REQUEST », « START_ACKNOWLEDGE » suivant une procédure qui est décrite en détail dans la norme IEEE 1355. Le but de cette procédure est de s'assurer que la transmission fonctionne. Ensuite, les blocs d'interface série déclarent le fonctionnement de leur liaison série au cœur R3P du routeur. Dans le cas où la calibration est perdue en cours de fonctionnement, ces mêmes automates vont relancer automatiquement la même procédure pour rétablir la communication.

3-4 Conclusion

Nous avons spécifié l'élément de commutation de notre réseau d'interconnexion : le routeur RCube. Le routeur RCube est un composant polyvalent, à haute performance. Il n'impose pas une topologie de réseau particulière. Il possède une forte bande passante grâce à l'utilisation de la liaison série 1 Gbit par seconde HSL. Sa latence est très fortement réduite par la mise en place du routage wormhole. L'utilisation du routage par intervalles diminue la taille des tables de routage, et un mode de destruction de l'en-tête permet de mettre en œuvre la technique de routage explicite. RCube possède par ailleurs une technique générale d'adaptabilité.

Le composant RCube utilise la liaison IEEE 1355. Une séparation entre le cœur du composant de routage, et le sérialiseur-désérialiseur 1355 permet d'intégrer rapidement d'éventuelles évolutions de celui-ci. Cette séparation permet aussi, éventuellement, d'utiliser d'autres types de liens physiques. Un bloc d'interface, distinct donc du cœur de routage, gère donc ce qui est propre au standard 1355. Une file d'attente asynchrone permet de s'affranchir des contraintes dues aux différentes horloges de fonctionnement des intervenants qui communiquent. Les mécanismes matériels mis en œuvre pour la sélection des sorties en utilisant le routage par intervalles et pour l'adaptabilité sont d'une très grande régularité, et d'une très grande simplicité. Le routage est donc effectué en quatre cycles.

Son implémentation matérielle est terminée et décrite en détail dans la thèse de Vincent Reibaldi. RCube a été réalisé en technologie 0,5 µm. RCube comporte 460.000 transistors. Le temps de cycle obtenu par simulation est de 12 ns, démontrant partiellement la validité de nos choix d'architecture. Les tests des circuits revenant de fabrication sont prévus au printemps 1996.

demandent la même sortie, ils l'obtiennent successivement. Sur la figure 3-p, on peut voir l'état du sélecteur de priorité de l'allocateur de sortie après des routages successifs de paquets.

Si les paquets sont de taille bornée, on peut donc borner le temps de cheminement maximum d'un paquet dans un réseau de diamètre *D*. Il faut en effet souvent que le délai de transmission soit borné pour garantir le fonctionnement de protocoles de plus haut niveau.

3-3-6 Gestion des erreurs.

Un certain nombre d'erreurs sont possibles sur les liens HSL. Nous les énumérons et précisons, le cas échéant, les traitements qui sont assurés par le routeur :

- Perte de calibration sur un lien série : un lien HSL tombe soudainement en panne. La fin des paquets dont la transmission vers ce lien était en cours, est détruite. Les paquets en provenance de ce lien, et dont la fin est indisponible, sont clos par le routeur, qui leur rajoute un caractère spécial « EEP » pour « Exceptional End of Packet ». D'un point de vue du routage, ce caractère est strictement équivalent à « EP ». Ce caractère sert à informer le récepteur du paquet que celui-ci a été tronqué en cours de routage.
- Erreur de parité sur un octet de l'en-tête : le paquet est détruit par le routeur, qui ne peut pas connaître avec certitude sa destination.
- Erreur de parité sur un caractère du corps d'un paquet : la fin du paquet est détruite en réception. En émission, le paquet est clôs par un caractère « EEP ».
- Erreur de parité sur un caractère « EP ». Lorsque RCube détecte un caractère EP présentant une erreur de parité, deux cas peuvent se présenter. Soit ce caractère était effectivement un EP, dont le bit de parité a été altéré, soit il s'agissait d'un caractère autre, que l'altération d'un bit a fait ressembler à un EP. Soit nous considérons ces caractères EP altérés comme des EP, soit comme des caractères altérés. Le premier choix implique que des paquets peuvent être scindés en deux, ce qui entraine la création d'un paquet avec un en-tête quelconque, peut-être invalide, qui peut générer des interblocages de routage wormhole. Le second choix implique qu'un paquet suivant celui dont le EP est altéré peut être détruit. Nous avons fait ce second choix.
- La perte complète d'un caractère n'est pas détectée par le routeur. En particulier dans le cas où des caractères de contrôle de flux sont perdus, et où des paquets sont bloqués, un redémarrage complet de tout le réseau d'interconnexion est nécessaire. Il sera détecté par l'interface et la correction sera assurée par les logiciels qui effectuent les échanges.

Notre philosophie sur les traitements d'erreur peut être résumé simplement : nous rejetons sur **l'interface matérielle** la détection et sur **les couches logicielles** le traitement des cas d'erreur qui auraient fortement compliqué la conception du routeur RCube ou qui en auraient ralenti le temps de cycle.

3-3-7 Reset.

La procédure d'initialisation du composant est relativement complexe. Elle nécessite obligatoirement un composant externe, un hôte, qui doit configurer le routeur RCube.

Tout d'abord, ce composant doit charger les tables de routage, comme nous l'avons vu ci-dessus. Un signal est alors donné au composant RCube qui démarre effectivement. Ensuite, les automates inclus dans les blocs d'interface série tentent de calibrer les liens. Pour cela, ils commencent à émettre des caractères « IDLE » pour calibrer le désérialiseur de leur

Ce mécanisme permet la régulation du flux, en limitant à 3% la perte de bande passante. Seul 1 caractère de contrôle de flux est systématiquement envoyé pour 32 caractères transmis effectivement, et ce quelle que soit la charge des liens.

Les caractères « FCC » ne peuvent être écrits dans la file d'attente, comme nous l'avons dit précédemment, ils sont donc resynchronisés par un opérateur annexe. La figure 3-o présente de manière schématique ce mécanisme.

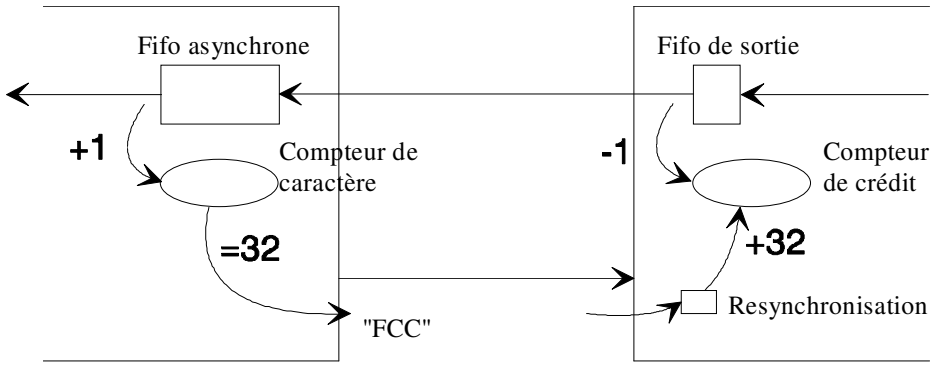

Figure 3-o : mécanisme de contrôle de flux.

3-3-5 Adaptabilité et équité.

La fonction d'adaptabilité est gérée en un seul cycle, comme nous l'avons vu, par le bloc de choix de sortie. Le registre qui code l'ordre dans lequel l'automate doit choisir les sorties ne permet pas de spécifier un ordre quelconque. En effet, un registre code 2^8 possibilités, alors qu'il existe *8!* ordres possibles. Le mécanisme exact de sélection est décrit dans la thèse de Vincent Reibaldi, mais on peut déjà dire qu'il permet de spécifier au moins une sortie prioritaire, et un jeu de sorties alternatives. Ceci reflète souvent la réalité des topologies, c'est-à-dire qu'il existe un « plus court » ou « meilleur » chemin et d'autres un peu plus longs, mais acceptables quand même.

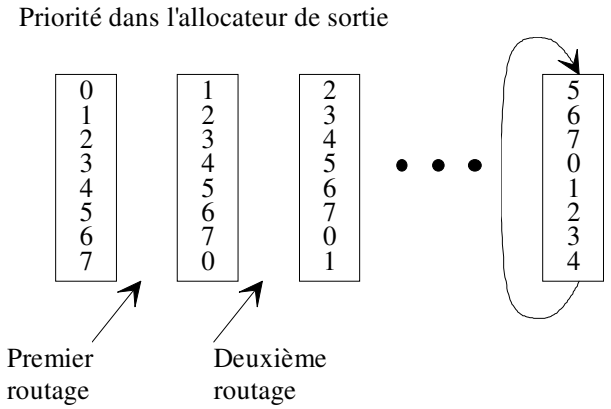

Figure 3-p : le sélecteur de priorité tournante.

L'autre enjeu, en cas de contention sur le réseau, est d'assurer l'équité entre les requêtes, quel que soit leur port d'entrée, tout en évitant les phénomènes de famine. Pour cela, les allocateurs de sorties possèdent un sélecteur de priorité tournante qui est déclenché par l'allocation du lien. Ceci permet d'assurer un routage équitable, c'est-à-dire que si tous les ports d'entrée

caractères « IDLE » par paquet, au moins, dans le cas de routage sans contention, voir figure 3-n).

Fin de paquet.

A la réception d'un caractère « EP » marquant la fin du paquet, l'automate d'entrée, après avoir transmis ce caractère, demande à l'allocateur de sortie la libération du lien. Celui-ci devient donc disponible pour le routage d'autres paquets.

Nous venons de décrire succinctement le synoptique d'une opération de routage dans RCube. Nous allons maintenant présenter les fonctionnalités annexes de ce composant.

3-3-3 Configuration.

La configuration du composant, c'est-à-dire le chargement des tables de routage, est simplifiée au maximum. Au démarrage du composant, un hôte externe inscrit ces tables au moyen d'un « scan-path » de configuration de largeur 8 bits. Cette solution, qui répond à des besoins pratiques de réalisation, a l'avantage de simplifier l'architecture du composant.

Les tables de routage des différents routeurs constituant un réseau définissent un état global : il est difficile de penser pouvoir les modifier dynamiquement. Le routage wormhole, la déséquentialisation des paquets dans le réseau, rendent impensable une reconfiguration d'un RCube, indépendamment des autres routeurs avec lesquels il cohabite. Les RCube d'un système seront donc configurés lors d'un rendez-vous global de tous les intervenants du réseau.

La configuration de RCube se résume à l'installation des tables de routage, de la programmation du format des en-têtes, et du mécanisme de « Header Deletion ». Toutes ces informations sont rangées dans des registres de 8 bits.

La configuration est effectuée via un « scan-path » parallèle 8 bits qui couvre la totalité de ces registres. A son extrémité, une sortie est prévue, de manière à envisager le chaînage des RCube. Pour plus de détails sur cette procédure, on se référera à la spécification de RCube, dans l'annexe 2.

3-3-4 Contrôle de flux.

Le contrôle de flux est assuré pour chaque port d'entrée-sortie par un automate indépendant. Celui-ci maintient deux compteurs :

- Le compteur des caractères extraits. En réception, ce compteur est incrémenté à chaque extraction d'un caractère de la file d'attente asynchrone de réception. Lorsque ce compteur arrive à 32, taille d'un « flit », un caractère « FCC » est inséré dans le flux de données sortant, et le compteur est remis à zéro.
- Le compteur de crédit : en émission, il est incrémenté de 32 par la réception d'un caractère « FCC », et décrémenté par l'émission d'un caractère. L'émission n'est autorisée que si le compteur de crédit est non nul.

Au démarrage, ces deux compteurs sont initialisés à zéro. Dans la procédure de démarrage, chaque composant IEEE 1355 transmet autant de caractères « FCC » qu'il a de « flits » disponibles dans sa file d'attente, pour en informer son partenaire. Il est ainsi possible de faire dialoguer entre eux des composants qui ont des tailles de files d'attente différentes.

Un registre de 8 bits code les 8 ports de sortie possibles pour un intervalle. Tout d'abord, deux mots de 8 bits représentant, l'un l'occupation des liens, l'autre le caractère défectueux des liens, masquent ce registre. En effet, un port peut être indisponible en cas de perte de calibration de la liaison physique.

Un deuxième registre, associé à chaque intervalle, code sur 8 bits un ordre de choix parmi les ports de sortie possibles. Un opérateur de priorité va permettre de sélectionner le lien libre et fonctionnel de plus forte priorité. Pour comprendre le fonctionnement détaillé de cet opérateur, on se référera à la thèse de Vincent Reibaldi[31] .

Dans le cas où aucun lien n'est disponible, l'automate d'entrée attend que l'un d'entre eux se libère. Le choix de la sortie réalise, en pratique, la fonction d'adaptabilité. Cette phase ne prend qu'un seul cycle.

Allocation des sorties.

A l'issue de ce premier arbitrage, il se peut que deux entrées demandent en même temps une même sortie. Un allocateur de lien effectue donc un choix, et sélectionne l'une d'entre elles. Cet opérateur est du même type que l'électeur de priorité évoqué ci-dessus.

Les allocations ne s'effectuent qu'aux cycles pairs de fonctionnement du composant, cette phase peut prendre un ou deux cycles.

Emission.

L'allocateur de lien pilote le « crossbar » pour établir le chemin entre l'entrée et la sortie. Les données sont alors transférées vers la file d'attente de sortie, d'où elles seront émises. Cette file d'attente, de quatre caractères de profondeur, ne sert qu'à amortir les contentions de flux sur la sortie. En effet, lorsque le lien de sortie ne peut plus émettre, la contention doit être rétro-propagée à travers le crossbar jusque l'entrée associée au transfert. La structure en pipe-line des opérateurs de routage implique donc la présence de cette file d'attente de sortie pour permettre de vider les caractères déjà extraits de la file d'attente d'entrée, et présents dans nos opérateurs.

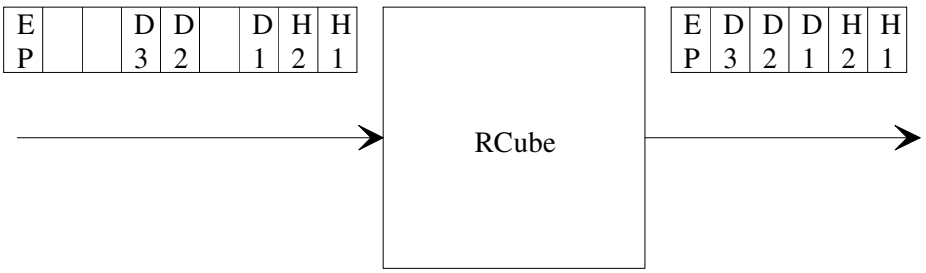

Figure 3-n : compactage des paquets.

Il est à noter que quatre cycles sont effectivement nécessaires pour analyser l'en-tête d'un message et prendre la décision de routage, dans le cas de non-contention. Par contre, la latence totale est de 10 à 11 cycles, à cause du temps de traitement dans les blocs sérialiseurs-désérialiseurs et de conversion 9 bits / 12 bits.

Le routeur, à cause de ces quatre cycles de délai pour le traitement de l'en-tête, est donc capable de recompacter des paquets qui présenteraient un flux légèrement haché (quatre

[31] [REIBALDI]

d'attente permet de ne pas séquentialiser le retour de ces caractères de contrôle de flux et le flot de données.

Choix de l'intervalle.

L'automate associé à chaque entrée du lien analyse l'en-tête du paquet arrivant. Dans un premier temps, il détermine à quel intervalle appartient la destination du message. Pour cela, il utilise un opérateur dont le synoptique est présenté sur la figure 3-l :

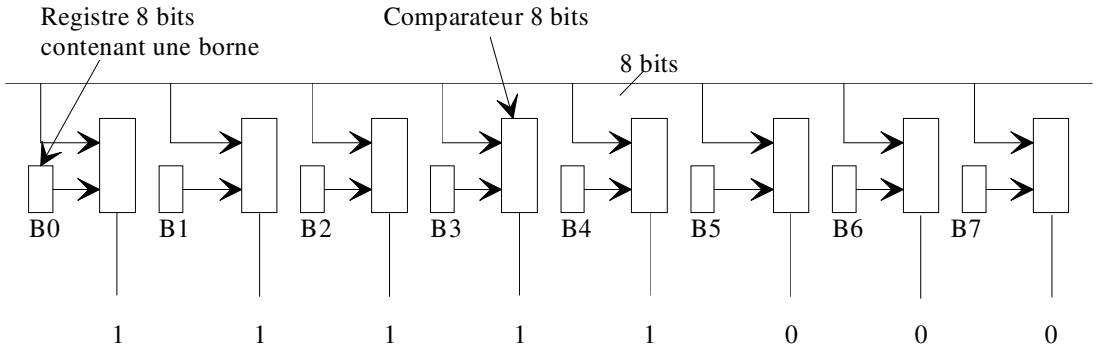

Figure 3-l : sélection de l'intervalle.

Un comparateur est associé à chaque registre qui contient une borne d'un intervalle. Ces dernières sont rangées par ordre croissant. Dans le cas des en-têtes sur un octet, RCube effectue les huit comparaisons en parallèle en même temps, comme représenté ci-dessus. L'unique transition de 1 à 0 des résultats de comparaison désigne l'intervalle. Dans le cas d'un en-tête sur deux octets, 2 cycles sont nécessaires. A l'aide de cette même circuiterie, la comparaison entre les poids forts de l'en-tête et des bornes est d'abord effectuée, puis celle des poids faibles.

Sélection d'un port de sortie.

Ensuite, l'automate utilise un second opérateur pour effectuer la demande d'un port de sortie.

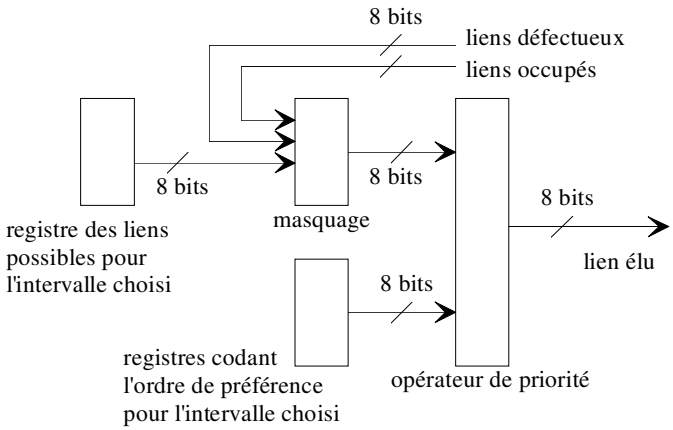

Figure 3-m : sélection d'un lien de sortie.

Il va choisir un port libre et fonctionnel, associé à l'intervalle décodé.

- Ecriture des caractères constituant les paquets dans la file d'attente asynchrone de réception, à la fréquence de l'horloge de réception.

Pour l'émission, il effectue globalement les opérations contraires :

- Codage des mots de 9 bits en mots de 12 bits.
- Insertion, lors de transferts hachés, de caractères « IDLE », pour garantir l'émission d'un caractère par cycle.

D'autre part, de manière globale, le bloc d'interface série implémente le protocole de démarrage des liens. En effet, une fois que la calibration du lien série a été détectée, quelques caractères de contrôle sont échangés pour garantir le fonctionnement du lien avant que le bloc d'interface série signale au routeur R3P que le lien est utilisable.

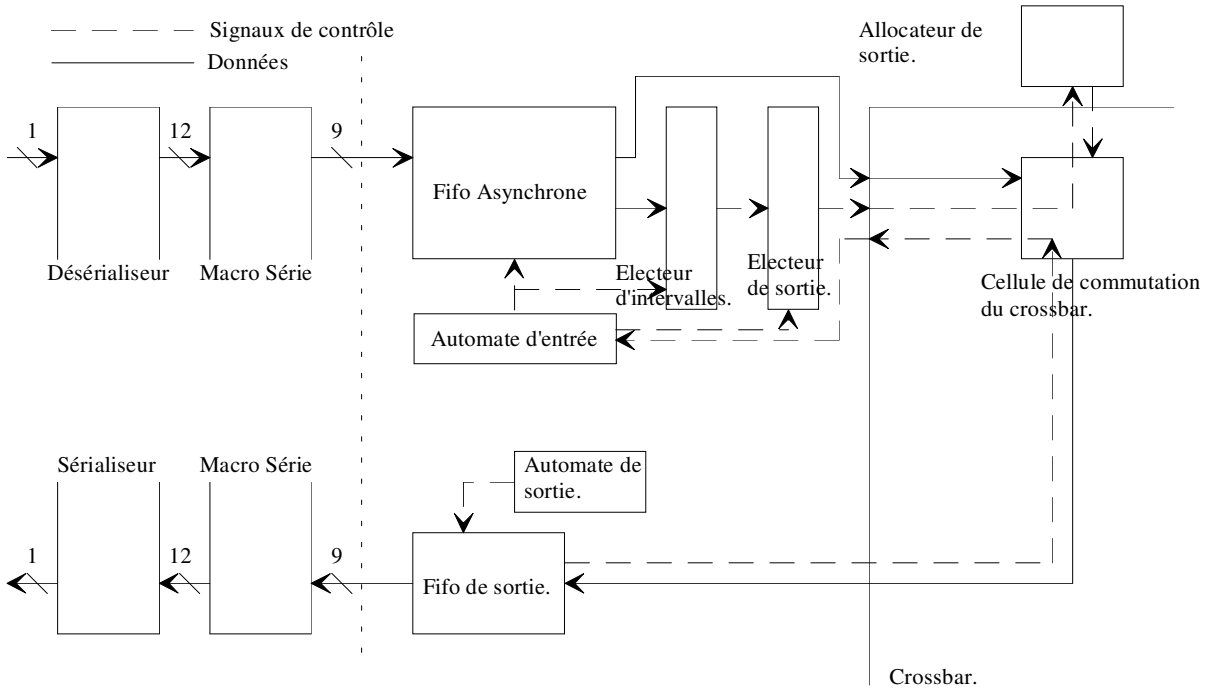

Figure 3-k : routage d'un paquet : interaction entre les blocs.

Synchronisation.

Le bloc d'interface écrit donc les caractères 9 bits dans la file d'attente asynchrone de réception. Les caractères de contrôle de flux ne sont pas inscrits dans la file d'attente asynchrone, mais resynchronisés avec l'horloge interne de R3P et signalés à l'automate de sortie associé au lien, pour que celui-ci puisse réguler son émission en fonction du crédit d'émission disponible.

La file d'attente asynchrone est un opérateur « full-custom », conçu au MASI. Elle permet l'écriture avec l'horloge récupérée sur le lien série HSL, et la lecture avec l'horloge interne. Sa fonction est double. Non seulement, elle permet la resynchronisation électrique des caractères reçus, mais elle sert aussi de mémoire tampon. Sa capacité est de deux « flits », soit 64 caractères.

Lorsqu'un « flit » est extrait de cette file d'attente pour être routé, un caractère de contrôle de flux est envoyé en sens inverse du flot de données. Cette contenance de deux flits dans la file

Le composant de routage RCube.

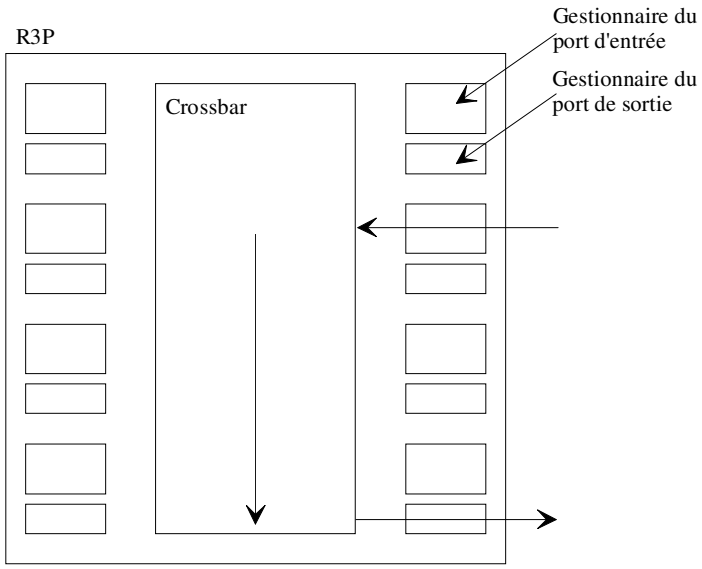

Figure 3-j : schéma bloc de R3P.

3-3-2 Synoptique d'un routage.

Sérialisation-désérialisation.

Tout d'abord, chaque caractère en provenance du lien série est désérialisé. Cette opération est effectuée avec une horloge récupérée sur le lien HSL et prend un ou deux cycles. La désérialisation nous fournit des caractères de 12 bits, tels que décrits dans le paragraphe 3-1-1, implémentant le « DC balance coding ». De même, à la sérialisation, nous devons fournir des caractères de 12 bits. L'émission se fait avec l'horloge interne du composant RCube. Chacune des opérations de sérialisation et de désérialisation s'effectue en 2 cycles.

Puisque le lien assure la transmission de l'horloge, il est indispensable de fournir à chaque cycle un caractère au sérialiseur; de même, le désérialiseur fournit à chaque cycle un caractère. La macro-cellule HSL fournie par BULL réalise de plus l'adaptation d'impédance du lien physique, la récupération des horloges et la détection de perte de la calibration (désynchronisation de l'horloge de réception).

Conversion 9 bits / 12 bits.

Le bloc d'interface entre R3P et le sérialiseur-désérialiseur remplit de nombreuses fonctions. Tout d'abord, en réception il effectue les traitements suivants :

- Décodage des mots de 12 bits en mots de 9 bits directement interprétables par le routeur R3P.
- Vérification de la validité de la parité des mots reçus.
- Extraction du flux de données des caractères « IDLE », qui servent uniquement à la conservation de la calibration du lien.
- Extraction et resynchronisation des caractères de contrôle de flux « FCC » avec l'horloge interne de R3P. En effet, ces caractères, bien qu'utilisés par R3P, ne peuvent être inscrits

3-3 Architecture interne.

3-3-1 Architecture générale.

Le partenariat avec Bull, qui fournit la macro-cellule sérialiseur-désérialiseur 1 Gbit / sec, a imposé un certain découpage dans l'architecture du composant RCube.

Nous avons défini deux sous-cellules :

- Un routeur à interface parallèle, R3P, présentant une interface normalisée pour chacun des 8 ports bidirectionnels. Cette interface parallèle de largeur un octet sera aussi utilisée dans le circuit d'interface décrit dans le chapitre 4.
- Un bloc d'interface série, qui permet de connecter cette interface parallèle aux sérialiseurs-désérialiseurs HSL. Cette conception modulaire permet, en effet, de bénéficier dans le futur de nouvelles évolutions de la liaison série sans avoir à effectuer une totale refonte du composant.

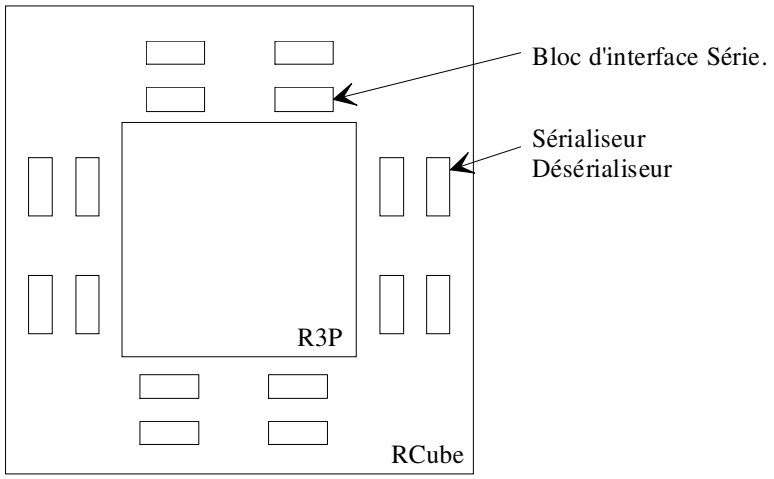

Figure 3-i : schéma bloc de RCube.

La fonction de routage est décentralisée. C'est-à-dire que chaque port d'entrée possède son propre automate, et ses propres tables de routage. Ces automates partagent le contrôle d'un opérateur global qui est le crossbar. La figure 3-j présente un schéma de l'architecture interne de R3P.

La procédure complète de routage d'un paquet s'effectue en une dizaine de cycles. La figure 3-k présente l'interaction entre les opérateurs impliqués dans le routage d'un paquet. Les paragraphes suivants explicitent plus précisément les opérations que le composant effectue à chaque cycle.

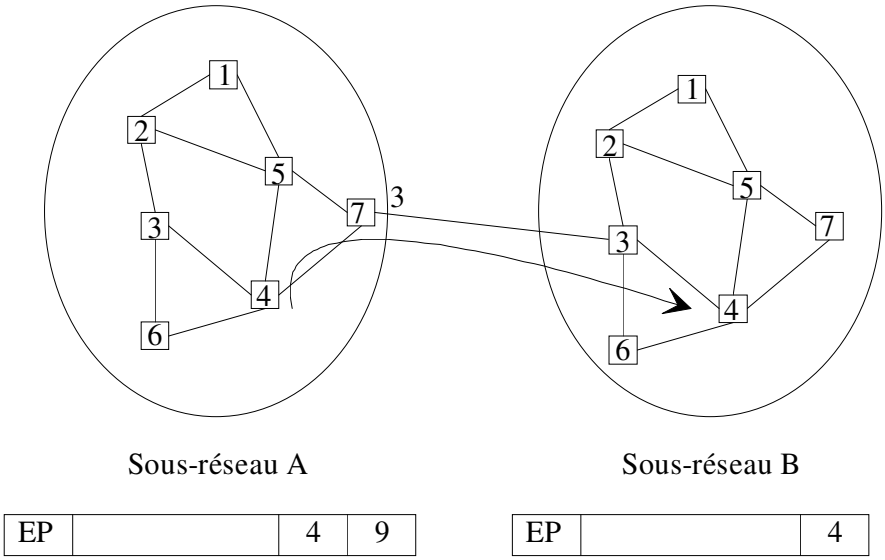

Figure 3-h : suppression de l'en-tête et sous-réseaux.

3-2-6 Adaptabilité.

Dans les cas de fort trafic, des phénomènes de « point chauds » peuvent apparaître. C'est-à-dire que certains liens du réseau sont demandés par plusieurs paquets et sont donc provisoirement bloqués. Pour contrer cela, nous mettons en œuvre une technique d'adaptabilité, qui permet de définir des chemins alternatifs en cas de contention.

A chacun des intervalles présentés au paragraphe 3-2-3, est associé un ensemble ordonné de sorties, et non une seule sortie. C'est-à-dire que pour chaque intervalle, la configuration du réseau peut définir plusieurs sorties. Ceci permet d'éviter les congestions non seulement dans le cas de topologies régulières, où il existe plusieurs chemins de même longueur entre deux points, mais même dans le cas où les chemins alternatifs définis sont de longueur supérieure au chemin nominal. C'est une généralisation du mécanisme de « Grouping » du routeur C104, présenté dans le chapitre 2.

Pour chaque paquet qui arrive dans un RCube, l'automate associé à son entrée choisira la sortie fonctionnelle et non occupée qui présente la plus grande priorité, parmi celles associées à l'intervalle contenant l'en-tête. Ce mécanisme permet aussi d'obtenir une plus grande disponibilité pour le système, dans la mesure où certains liens défectueux peuvent être contournés dynamiquement, sans reconfiguration du système.

Il faut cependant noter que l'utilisation de ce mécanisme introduit une éventuelle déséquentialisation des paquets dans le réseau : ils ne sont plus forcément reçus dans l'ordre où ils ont été émis.

Dans la section suivante, nous présentons l'architecture interne du routeur RCube. Il est à noter que ce travail a été effectué en étroite collaboration avec Vincent Reibaldi et Belkacem Zerrouk. Vincent Reibaldi a par ailleurs conçu la micro-architecture des opérateurs optimisés nécessaires pour atteindre la fréquence visée et effectué l'implantation physique.

pouvaient se produire pour différentes topologies à cause du routage wormhole, et proposé des solutions de marquages pour les éviter.

La technique de double mémorisation présentée dans le chapitre 2 et présente sur le CRAY T3D permet de régler les problèmes d'interblocage de manière plus satisfaisante sur le plan théorique. Cependant le surcoût matériel, et surtout l'augmentation de la complexité des automates inclus au routeur ont rendu cette solution impraticable dans notre contexte. Là aussi, ces aspects pourront être reconsidérés ultérieurement.

Avec ce mécanisme, la taille des tables de routage est donc proportionnelle à la valence du routeur, c'est-à-dire au nombre de liens, ici 8. On voit sur la figure 3-g que 8 octets suffisent pour le codage des tables dans le cas des en-têtes à 1 octet, 16 pour celui des en-têtes à 2 octets. Ces 8 ou 16 octets définissent les 8 intervalles I0 - I7 associés aux 8 sorties.

3-2-4 Une table de routage par entrée.

Il existe une table de routage par lien d'entrée. Ceci permet tout d'abord d'expliciter la concurrence des accès aux tables de routage par chacun des automates d'entrée et présente deux avantages supplémentaires :

- Dans le cas d'un réseau, comme un tore par exemple, où il existe de nombreux chemins de même distance entre deux points, il est possible de répartir la charge, en répartissant les paquets le long de ces chemins en fonction de leur entrée dans leur routeur. C'est une répartition statique de la charge.
- Un autre avantage est la possibilité de partitionner un réseau composé de routeurs RCube, et ainsi d'obtenir deux sous-réseaux logiques, totalement séparés. Les paquets qui transitent dans l'un de ces réseaux sont ainsi garantis de ne jamais changer de sous-réseau. Ceci peut-être particulièrement intéressant dans certaines architectures où plusieurs classes de paquets, ayant des urgences distinctes, cohabitent.

3-2-5 Header Deletion.

Chaque port de sortie d'un routeur peut être configuré de façon à supprimer l'en-tête des paquets qui sont émis. Cette possibilité est réglée de manière statique à la configuration du composant.

La fonction de suppression de l'en-tête a une double portée. Tout d'abord, elle permet de dépasser la limitation à 65536 du nombre de processeurs, mais elle explicite aussi éventuellement la notion de sous-réseau. Dans la figure 3-h, on voit deux sous-réseaux, reliés par un seul lien.

Dans le premier sous-réseau A, on choisit que les paquets à destination du deuxième, B, portent l'en-tête « 9 ». Le lien 3 du routeur passerelle 7 est configuré en « Header Deletion » et son intervalle associé est réduit aux en-têtes « 9 ». Les paquets portent, en deuxième position, le numéro du processeur dans le sous-réseau B. Les paquets sont ainsi routés hiérarchiquement vers leur destinataire.

Si cette fonction est activée sur la totalité des liens, une autre méthode de routage peut alors être activée, c'est la méthode de routage explicite, où chaque paquet porte comme en-tête la liste des sorties dans chaque routeur qu'il doit traverser. Ce mode de fonctionnement peut présenter un certain intérêt, en particulier pour la mise au point d'un système.

Le composant de routage RCube.

Entre deux routeurs, le contrôle de flux est assuré avec une granularité de 32 octets, en revanche, à l'intérieur d'un RCube, la taille quelconque des paquets impose que le contrôle de flux entre l'entrée et la sortie se fasse caractère par caractère.

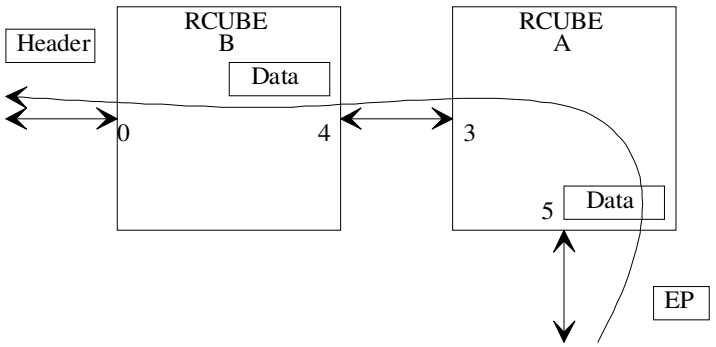

Figure 3-f : routage wormhole.

3-2-3 Routage par intervalles.

Le routeur RCube est un routeur générique. C'est-à-dire que nous ne faisons aucune hypothèse sur la topologie du réseau. Dans chaque RCube, une table de routage contient les correspondances entre l'en-tête et le lien de sortie. Cette association ne peut se faire directement, car dans le cas d'un en-tête sur deux octets, 65536 entrées seraient nécessaires dans cette table.

On utilise donc une technique de routage « par intervalles »[30], similaire à celle utilisée par INMOS et présentée dans le chapitre 2.

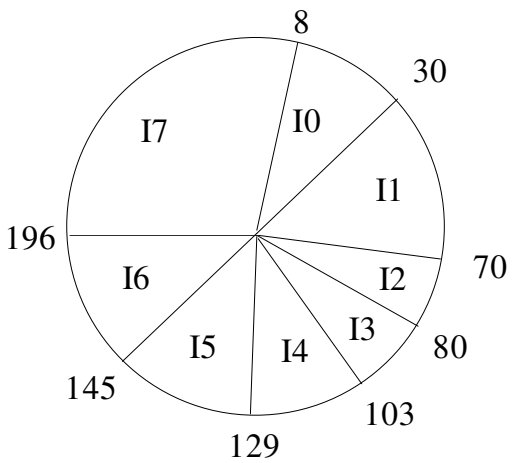

Figure 3-g : le routage par intervalles.

On associe à chaque nœud du réseau un numéro, et à chaque sortie d'un routeur un intervalle. Lorsqu'un paquet se présente, on compare son en-tête avec les différents intervalles des sorties pour déterminer dans lequel l'en tête se trouve, et le paquet est ainsi routé.

Van Leeuwen a prouvé que, quel que soit le réseau, il existe un marquage des processeurs, et une association d'intervalles aux sorties qui permettent d'amener les messages à bon port. Belkacem Zerrouk a d'autre part caractérisé les éventuels problèmes d'interblocage qui

[30] [LEEUWEN87], [LEEUWEN91], [ZERR93] et [ZERR94]

génération, prévue pour l'an prochain, devra considérer plus en détail l'implémentation des canaux virtuels dans un routeur wormhole.

3-2 Spécification.

3-2-1 Présentation de RCube.

RCube est un routeur de paquets. Du point de vue fonctionnel, il s'inspire fortement du routeur C104. 8 liens bidirectionnels sont pleinement interconnectés. C'est-à-dire qu'en même temps, 8 paquets peuvent transiter, entre 8 entrées et 8 sorties.

Il est capable de prendre des décisions simultanées sur chacune de ces entrées. C'est-à-dire qu'il existe 8 automates différents qui analysent les en-têtes de chaque paquet se présentant sur les 8 entrées.

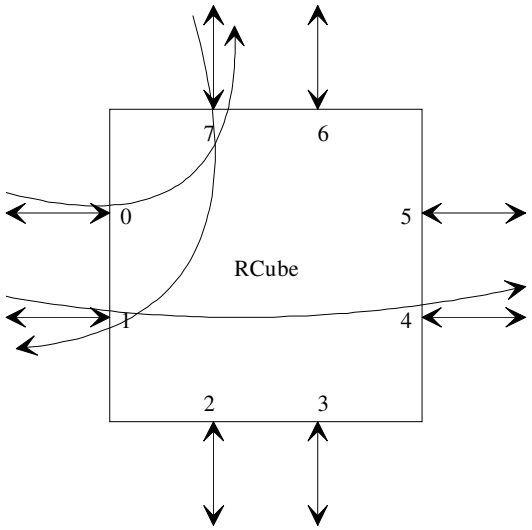

figure 3-e : le routeur RCube.

3-2-2 Contrôle de flux.

La technique de routage wormhole impose aussi la mise en place d'un dispositif de contrôle de flux. Si par exemple, sur la figure 3-f qui présente le routage wormhole, le lien 0 du RCube B est occupé par d'autres transmissions, il faut rétro-propager cette contention au lien 3 puis 5 du RCube A.

Pour cela un caractère de contrôle de flux « FCC », (caractère spécial de 9 bits), est inséré en sens inverse du flot de données pour autoriser l'émission d'un « flit » (une rafale) de 32 caractères. C'est-à-dire qu'à chaque fois que 32 caractères ont été transmis depuis l'entrée 4 vers la sortie 0 du RCube B, un caractère FCC est transmis vers la sortie 3 du RCube A.

Chaque port d'entrée possède une file d'attente de deux « flits » complets (64 octets), qui permet de ne pas séquentialiser le retour des FCC avec le flot de données. La génération du caractère « FCC », sa transmission dans le réseau et son traitement prennent quelques cycles. Pour que la transmission puisse continuer parallèlement à ces opérations, la file d'attente de réception doit comprendre plus d'un « flit ». Avec deux « flits », le débit est continu pour des liaisons point-à-point d'une longueur inférieure à une vingtaine de mètres.

peut la résumer en quelques mots : lorsqu'un cycle est présent dans le réseau, on multiplie, comme dans le cas du T3D, le nombre d'éléments de mémorisation aux entrées des liens pour éviter les interblocages. Chacun de ces éléments de mémorisation correspond à un canal virtuel.

Certaines topologies, comme le CRAY T3D, requièrent 2 canaux virtuels pour éviter les interblocages, d'autres, comme le « Cube Connected Cycle » en nécessitent trois[27] .

Pour éviter les interblocages, il est aussi possible de doubler les liens de communication plutôt que les éléments de mémorisation interne. Cette solution est celle que nous avons retenue. L'un des arguments est que le nombre de canaux virtuels est fixé à la construction du routeur, et que nous voulons que notre composant puisse supporter n'importe quel type de topologie. La présence de canaux virtuels n'aurait donc pas empêché le doublement de certaines connexions entre des routeurs pour des topologies particulières, même si elle en aurait diminué le nombre.

La congestion.

Le deuxième inconvénient du routage «wormhole » est qu'en cas de contention sur une ressource, lorsqu'un paquet est bloqué lors de son routage, la contention se propage dans le réseau, car les paquets derrière celui qui est bloqué ne peuvent avancer. Pour cette raison, il est nécessaire que les liens d'entrée des routeurs disposent d'une mémorisation importante. Pour augmenter l'efficacité de cette mémorisation, le SP2 propose un mécanisme « Buffered Wormhole Routing », qui met la mémorisation en commun entre les liens d'entrée, ce qui est plus efficace.

Le « virtual cut through »[28] , propose une autre solution, en n'autorisant le début du transfert d'un paquet vers un routeur que lorsque les files d'attente de réception disposent de suffisamment de place pour contenir la totalité du paquet. Cette technique impose une limite de la taille des paquets qui est incompatible avec nos objectifs.

Une troisième technique, le « Virtual Channel Flow Control »[29] permet d'atténuer les phénomènes de contention sur le réseau, par la multiplication des files d'attente d'entrée de chaque routeur et autorise le multiplexage temporel de différents paquets sur un lien. La charge que peut supporter le réseau augmente alors régulièrement avec le nombre d'éléments de mémorisation que l'on ajoute.

Conclusion.

La technique des canaux virtuels est donc particulièrement performante. On peut, en fait, la résumer de la manière suivante : **on partage un lien physique entre plusieurs ports**. Ce mécanisme permet d'augmenter la fraction utilisable de la bande passante du lien et de résoudre les conditions d'interblocage.

Les relations entre l'utilisation des canaux virtuels et la technique de routage par intervalles (voir section 3-2-3) sont étudiées en détail dans [ZERR94]. Les contraintes d'implémentations de RCube, en particulier le temps de cycle et la complexité des automates de routage, nous ont amenés à écarter provisoirement cette solution, dans la mesure où cela ne constituait pas un des objectifs principaux de la présente réalisation. Une étude sur un routeur de seconde

[27] [DALLY87]
[28] [KERMANI]
[29] [DALLY90]

- La granularité 8 bits permet de transporter simplement des données qui ne sont pas alignées sur des frontières de mots de 32 ou 64 bits. Notre composant d'interface, présenté dans le chapitre suivant, utilise avantageusement cette particularité.
- Les systèmes imposant des paquets de taille fixe « gâchent » plus de bande passante que ceux où la taille des paquets est quelconque. A moins que la taille des transferts soit exactement multiple de la taille des paquets, un certain nombre de paquets contenant des données inutiles sont transmis par les systèmes à taille de paquets fixe.
- Le paquet peut être de taille infinie, ceci équivaut à l'établissement d'une liaison permanente entre un émetteur et un récepteur. Le caractère EP, pour « End of Packet », marque la fin du paquet.

Un réseau de RCube est donc un médium de transport de données. Son format de paquet est extrêmement souple. La figure 3-d présente de manière synthétique la succession des couches de protocoles dans un système, et y place le format des paquets du routeur RCube.

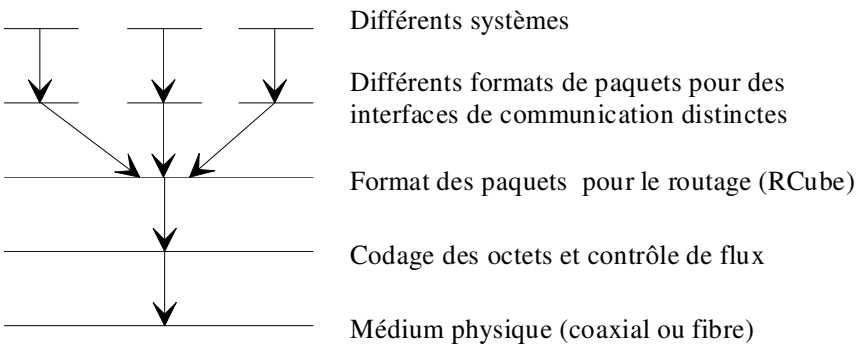

Figure 3-d : Protocoles.

3-1-3 Routage Wormhole.

L'une des principales préoccupations qui nous guide dans la conception du routeur RCube, est la latence, qui doit être aussi faible que possible.

La technique de routage « Wormhole », où les paquets sont routés vers un lien de sortie, après analyse de leur en-tête et avant la réception complète du message, permet de diminuer la latence totale du routage, qui est plus faible que dans le cas des routages « store-and-forward ». D'autre part, seule une technique de ce type peut s'accommoder de paquets de taille quelconque.

La technique de routage « Wormhole » présente deux inconvénients majeurs, qu'il convient de détailler.

L'interblocage.

Le paragraphe 2-1-1 présente une situation dans laquelle un interblocage aurait pu apparaitre, ainsi que la manière dont le routeur du T3D l'a évité. Ce mécanisme est un cas particulier des « virtual channel » de Dally[26] .

Lorsque la topologie du réseau d'interconnexion est fixée, Dally propose un mécanisme qui permet d'assurer que des conditions d'interblocage ne peuvent apparaître dans le réseau. On

[26] [DALLY87]

Les caractères dont les nombres de « un » et de « zéro » sont égaux peuvent être de deux types : soit des caractères de données normaux, avec leur 9 ème bit à zéro, soit des caractères spéciaux de contrôle, dont le 9 ème bit est à un. Leur différentiation s'effectue grâce au choix de bit « INV ». Les caractères de contrôle ne sont pas inversés alors que ceux de données le sont. Un certain nombre de ces caractères de contrôle sont utilisés dans la norme IEEE 1355 pour le contrôle de flux, le démarrage de la liaison et diverses signalisations.

La figure 3-b présente de manière synthétique les cas où l'octet à transmettre est inversé lors de son transport effectif.

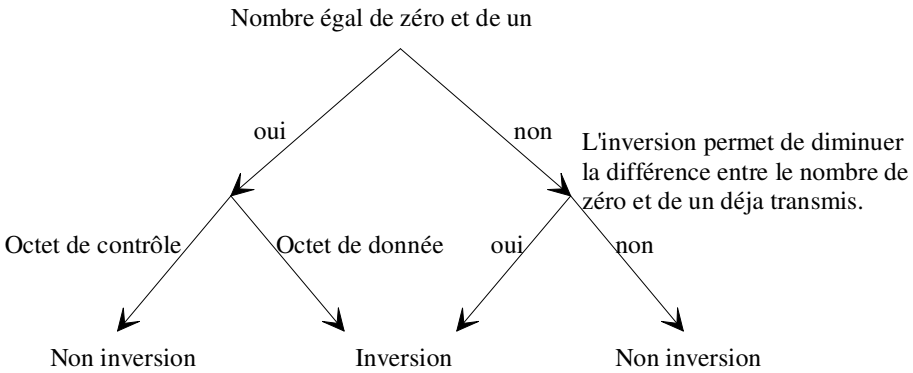

Figure 3-b : Inversion.

L'un des caractères de contrôle « IDLE » a un rôle assez particulier. En effet, l'horloge d'échantillonnage en réception est extraite du signal électrique. Un émetteur HSL est donc tenu de transmettre à chaque cycle un caractère, pour que l'extracteur d'horloge du récepteur reste calibré. Le caractère « IDLE » est donc systématiquement émis à chaque cycle où des données utiles ne sont pas effectivement transmises.

3-1-2 Paquets de taille quelconque.

Le format des paquets est présenté sur la figure 3-c. Les paquets sont de taille quelconque, en octets. Il existe deux formats de paquets suivant la taille de l'en-tête.

Les en-têtes à un octet permettent d'adresser 256 destinataires, ceux à en-tête de deux octets, 65536 destinataires. Les circuits de routage peuvent être configurés dans l'un ou l'autre des deux modes de façon indépendante pour chacun des ports d'entrée.

Figure 3-c : format du paquet RCube.

Les avantages de n'avoir aucune contrainte sur la taille du paquet sont nombreux :

- N'importe quel type de transfert peut être effectué par un réseau de routeur RCube. Un réseau de RCube peut donc être considéré comme indépendant des protocoles qu'il véhicule.

Chapitre 3 : Le composant de routage RCube.

Nous étudions dans ce chapitre le premier des deux éléments qui constitue notre réseau d'interconnexion, c'est-à-dire l'élément de commutation.

3-1 Motivation.

L'une des motivations principales qui a guidé la spécification du composant RCube est la généricité. Une utilisation la plus large possible du routeur RCube doit être possible dans différents contextes de machines parallèles. D'autre part, il doit permettre le routage de données dans les meilleures conditions de latence.

Il s'agit surtout de développer un élément de routage intégré en un seul composant pour des raisons de coût et de simplicité, tout en proposant les meilleures performances possibles.

Il s'agit enfin de mettre en valeur la technologie HSL, particulièrement bien adaptée à ce genre de réalisation, dans la mesure où elle propose, sur un support matériel simple et en technologie CMOS, un débit important (1 Gbits / sec).

3-1-1 La liaison point-à-point HSL.

Le standard IEEE 1355 spécifie, entre autres, le fonctionnement de la liaison série HSL à 1 Gbit / sec. Dans la suite de ce document, nous assimilerons IEEE 1355 et la liaison HSL.

La liaison HSL permet de transférer des caractères codés sur 9 bits. Le 9 ^{ème} bit permet la distinction entre les données et les caractères de contrôle, qui sont ainsi facilement isolables du flux de données. En fait, lors de leur transfert effectif sur le lien de transmission, les caractères sont codés sur 12 bits. La transformation 9 bits vers 12 bits est décrite sur la figure 3-a :

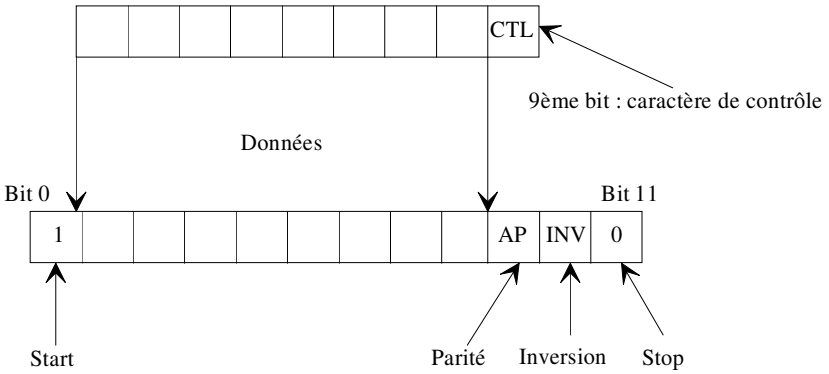

Figure 3-a : codage 9 bits vers 12 bits.

Les bits « START » et « STOP » garantissent la présence d'un front dans chaque caractère transmis. Ce front est utilisé pour la récupération de l'horloge d'émission par le récepteur.

Sur le signal transmis par le lien série, la composante électrique continue doit être nulle, c'est-à-dire que les nombres de « zéro » et de « un » transmis doivent être sensiblement identiques. Ainsi, l'émetteur, à la sérialisation, décide parfois d'inverser les bits de certains caractères. Dans le codage sur 12 bits, le bit INV indique si l'inversion a eu lieu et sert au récepteur pour reconstruire le bon mot de 8 bits. Le bit AP code la parité de l'ensemble des données et du bit INV.

bus MCA, qui est un peu lent par rapport au débit des liens du système SP2 d'une part, et aux besoins de certaines applications d'autre part.

Le principal inconvénient est le coût. On peut remarquer que la conception de la carte réseau est du même ordre de complexité qu'une cellule processeur de carte mère complète, si ce n'est que le processeur utilisé est d'une génération antérieure.

2-5 Conclusion.

Nous venons de présenter quatre systèmes d'interconnexion point-à-point différents. Chacun d'entre eux présente des solutions très positives à notre problème.

Le CRAY T3D propose un mécanisme de transfert de messages, en plus d'un mécanisme de partage de mémoire. Le processeur construit explicitement les paquets, par des accès dans une portion spécifique de l'espace d'adressage. En réception, les messages sont placés dans une file d'attente, d'où le processeur réceptionnaire les extrait. Ce système de communication est fortement couplé au processeur, et la topologie de la machine est figée par la conception du routeur.

Le système de communication Myrinet présente l'avantage de réduire le nombre de recopies des données entre les applications et le réseau pour l'envoi de messages, mais en laisse subsister. Il utilise pour cela des pages de la mémoire principale dédiées aux transferts. Il propose deux interfaces de communication, l'une par une API spécifique, l'autre par un portage optimisé de TCP / IP.

Le routeur C104 d'INMOS est une expression très aboutie de routeur de paquets wormhole à topologie quelconque. La combinaison du routage par intervalles et du routage explicite lui confère une grande souplesse. Le mécanisme de « Grouping » permet de contrebalancer le faible débit des liens, mais ne propose qu'une solution particulière au problème général de l'adaptabilité.

Le SP2 d'IBM propose un adaptateur réseau sur bus standard, possédant un processeur propre pour la gestion des protocoles. Celui-ci décharge grandement le processeur de certaines tâches. Il propose une interface sans appels systèmes, à faible latence, mais dont le débit est limité. Le coût de cette interface complexe est cependant assez élevé, à cause des nombreux composants, commerciaux et spécifiques, qui s'y trouvent.

Nous avons essayé de combiner les avantages de ces solutions, de proposer un mécanisme qui permette encore de réduire le nombre de recopies logicielles pour le transfert de messages, de spécifier un routeur générique présentant une vraie adaptabilité et un débit élevé, tout en ayant des composants à faible coût de développement et de mise en œuvre. Les trois chapitres suivants présentent notre solution, dont les performances sont comparées, dans le chapitre 6, à celles qui viennent d'être présentées ici.

vers la mémoire de l'adaptateur. Le processeur i860 initie alors un second transfert DMA, effectué par le deuxième contrôleur de DMA i860, qui transfère les données depuis la file d'attente vers le contrôleur de lien. Celui-ci émet alors les données sur le lien.

Réception de données.

Dans le cas de la réception de données, le i860 écrit dans la file d'attente un descripteur du DMA à accomplir par le contrôleur MCA, c'est-à-dire l'endroit dans la mémoire principale où il faut placer les données. Ensuite, le contrôleur de DMA i860 transfère les données depuis le contrôleur de lien vers la file d'attente, où elles sont ensuite envoyées sur le bus MCA par le contrôleur de DMA MCA.

Le processeur i860 assure aussi la vérification de l'intégrité des données et exécute un protocole permettant d'en redemander l'émission en cas d'erreur intervenue pendant la transmission.

Ce dispositif est fort complexe, on peut cependant en résumer ici les avantages principaux :

- Très faible latence : dans ce dispositif, il n'est pas nécessaire aux applications d'effectuer des appels systèmes car le mécanisme de protection de mémoire s'adapte aussi à la mémoire de l'adaptateur réseau. Ainsi le dialogue entre le processeur i860 qui gère le protocole et l'application est direct, et sans intermédiaire.
- Faible « Overhead » : la charge de traitement à accomplir par le RS6000 est faible. La plupart des opérations sont accomplies par le processeur i860 embarqué sur la carte d'interface.
- Protocole tolérant : le processeur i860 garantit que les messages demandés sont effectivement transférés. Il gère la totalité des protocoles permettant de détecter et de corriger les erreurs réseau.

Les performances de ce système, annoncées par IBM, sont fort intéressantes.

La latence complète d'un transfert de message est de 39 µs, en incluant les traitements logiciels effectués par le RS6000, la latence de l'adaptateur, et un étage de routage.

Le système est capable de transférer 24 Mo / sec., en émission et en réception simultanées, pour une bande passante globale de 48 Mo / sec. Le facteur limitant est le bus MCA, qui avec 77 Mo / sec. de transfert maximum en crête, pénalise le système.

2-4-5 En résumé.

Le système de communication du SP2 présente de très bonnes performances. Construit autour d'un routeur de paquets Vulcan, possédant 8 ports parallèles bidirectionnels, il permet le transfert de paquets avec une très faible latence. La technique de routage choisie, le routage explicite wormhole, est enrichie par un système de mémorisation interne qui permet de contrer une partie des effets négatifs de ce type de routage dans les cas de contentions.

L'adaptateur réseau s'adapte sur le bus MCA des nœuds de calcul, ce qui permet de supporter très rapidement les évolutions des microprocesseurs. Il contient un dispositif complexe orchestré par un processeur i860. Celui-ci prend en charge les protocoles d'émission et de réception de messages, ainsi que l'interface avec les processus émetteur et récepteur sans que ceux-ci aient à effectuer des appels systèmes.

La latence d'un transfert de message est très faible, 39 µs, tandis que la bande passante utilisable par chaque processeur reste assez réduite, avec 48 Mo / sec. Ceci est dû au choix du

- 8 Mo de mémoire : cette mémoire contient les programmes et les données que le i860 manipule. Elle permet en outre une mémorisation des paquets reçus ou à émettre. Elle est partagée par le RS6000, qui peut lui aussi y écrire et y lire des données.
- Un gestionnaire du lien et de la mémoire : ce composant gère le protocole du lien SP2. Il contient deux files d'attente de 2 KOctets, ainsi que la logique nécessaire à l'envoi et à la réception des données. Il gère aussi les 8 Mo de mémoire dynamique présente sur la carte.
- Un contrôleur de bus MCA : ce composant gère les signaux de contrôle propres au bus MCA.
- Une file d'attente bidirectionnelle : cette file d'attente est utilisée lors de transferts de messages. En émission, les données en provenance du bus MCA y sont écrites, puis lues par le i860; en réception, l'i860 y écrit des données, qui sont ensuite envoyées vers le bus MCA.
- Un contrôleur de DMA MCA : celui-ci assure la recopie des données entre la file d'attente bidirectionnelle et le contrôleur de bus MCA.
- Un contrôleur de DMA i860 : il assure le transfert des données entre la mémoire du i860 et la file d'attente bidirectionnelle.

La figure 2-u présente l'interaction entre ces différents éléments.

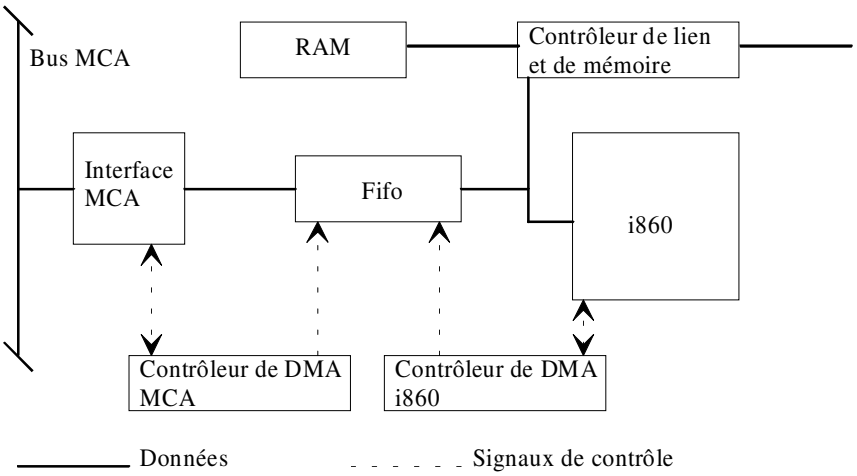

Figure 2-u : architecture du contrôleur MCA du SP2.

Un mécanisme complémentaire, non représenté sur le schéma ci-dessus, permet au RS6000 de lire et d'écrire dans la mémoire de la carte réseau. Ceci permet au processeur principal et au i860 de se synchroniser.

Emission de données.

Le processeur RS6000 écrit tout d'abord dans la mémoire de la carte réseau une structure contenant toutes les informations permettant de caractériser le message à envoyer, c'est-à-dire la destination, la taille etc. Ensuite, le reste de l'émission est réalisé par le i860. Celui-ci va en particulier assurer la segmentation des messages en paquets et créer les en-têtes.

Pour émettre des données vers le réseau, le processeur i860 écrit dans la file d'attente un descripteur du transfert DMA à accomplir. Ce descripteur identifie la zone de mémoire où se trouvent les données à émettre dans la mémoire centrale du système. Le contrôleur de DMA MCA effectue ce DMA, et transfère donc les données de la mémoire principale du nœud SP2

routage à très faible latence. Le temps entre la réception du premier octet du paquet et son émission est de 5 cycles soit 125 ns par routeur[25] .

Chaque port d'entrée possède une file d'attente pour permettre la mémorisation des données et le fonctionnement du contrôle de flux. Ces files d'attente sont de petite taille (31 octets). De plus, le composant Vulcan possède une mémoire interne de 1 KOctet pour mémoriser provisoirement les paquets dont le routage ne peut être effectué sur le champ. Cette technique s'appelle le « Buffered wormhole routing ». Dans le cas où un port de sortie est occupé, le paquet entrant est mémorisé dans cette mémoire interne, commune aux 8 ports. Dès que le port de sortie se libère, le paquet contenu dans cette mémoire est routé. Ce mécanisme permet de diminuer fortement les impacts négatifs des contentions sur le réseau. En effet, les réseaux « wormhole » sont très sensibles aux contentions, dans la mesure où un message bloqué peut en bloquer d'autres. Ici, ce phénomène est amoindri par l'utilisation de cette mémoire centrale, partagée par les 8 ports, qui permet de dégager certains des ports d'entrée de paquets qui ne peuvent être routés momentanément, pour permettre le routage de ceux qui le suivent. La figure 2-t montre un synoptique de l'architecture interne du routeur Vulcan.

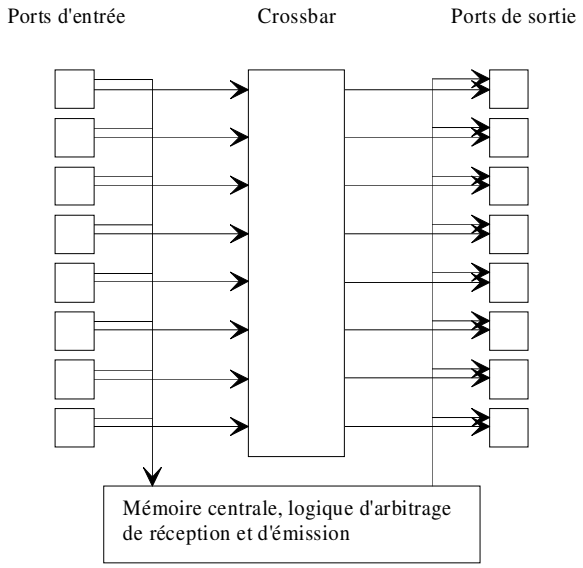

Figure 2-t : architecture interne du routeur Vulcan.

Le fait de centraliser la ressource mémoire, et d'en permettre un partage dynamique entre les ports, permet d'obtenir de meilleures performances que dans le cas où chacun des ports possède un peu de mémoire. On est donc sûr de pouvoir utiliser cette ressource au maximum, en l'allouant aux ports du composant qui en ont besoin.

2-4-4 Le système d'interface SP2.

Le système d'interface entre le bus MCA et le réseau d'interconnexion du SP2 est une carte complexe, composée de 7 éléments principaux :

- Un processeur i860 : celui-ci gère effectivement les protocoles de communication. Il met en forme les paquets, pilote le gestionnaire du lien et initie tous les transferts DMA.

[25] [STUNKEL]

de communication spécifique. Les nœuds sont reliés entre eux par un réseau de liaisons point-à-point parallèles et des routeurs. La figure 2-r présente l'architecture globale du SP2. Les routeurs du réseau sont regroupés sur une carte du système[24] .

2-4-2 Le lien de communication.

Ici encore le réseau de communication présente une technique de routage wormhole, avec un contrôle de flux.

L'une de ses particularités est que la totalité des liens du réseau d'interconnexion est cadencée par la même horloge à 40 Mhz, différente de celle des processeurs. La résolution des asynchronismes s'effectue donc dans le composant d'interface, entre la partie fonctionnant à la vitesse du bus MCA et celle pilotant le lien. Cette distribution d'une horloge centralisée, là encore, est une source de complexité pour le système. Des éléments de retard réglables sont présents sur chaque composant d'interface pour compenser les déphasages entre les deux horloges de deux composants s'échangeant des données. Le débit maximum de ce lien dans le SP2 est de 40 MOctets / seconde.

Les composants s'échangent des paquets. Un paquet est de taille quelconque mais limitée à 255 caractères de 8 bits. Le premier octet contient la taille du paquet, le, ou les suivants, les indications pour le routage, puis enfin les données.

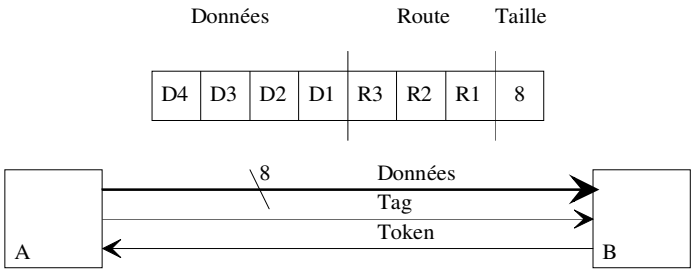

Figure 2-s : transfert de paquets.

Le signal « Tag » sert à spécifier qu'un caractère de données est effectivement présent sur le bus. Le contrôle de flux s'effectue sur la base d'un caractère. A chaque caractère traité par le récepteur du paquet, c'est-à-dire extrait de la file d'attente de réception, un bit « token » est envoyé sur le fil prévu à cet effet, en sens inverse du flot de données.

2-4-3 Le routeur Vulcan.

Le routeur Vulcan est un routeur de paquets à 8 liens bidirectionnels contenant un crossbar totalement interconnecté. Il est donc capable de router simultanément 8 paquets, depuis 8 ports d'entrée vers 8 ports de sortie.

La technique de routage adoptée est le routage explicite. C'est-à-dire que l'en-tête comporte la liste des ports de sortie successifs dans chaque routeur à traverser. Chaque octet de l'en-tête comporte deux champs de 3 bits désignant une sortie, et deux bits pour savoir lequel de ces deux champs il faut utiliser. Une fois que les deux entrées ont été utilisées, le caractère est détruit, et la taille du paquet décrémentée. Cette procédure fort simple permet d'obtenir un

[24] [AGER]

paquets puissent automatiquement prendre l'un ou l'autre. Finalement ce mécanisme ne sert qu'à compenser le faible débit de la liaison DS-Link, 10 MOctets / sec.

2-3-5 En résumé.

Le routeur C104 présente des solutions techniques intéressantes. La liaison série point-à-point permet la diminution du coût des systèmes par rapport aux liens parallèles. Le DS-LINK présente un contrôle de flux, dont le coût reste constant quelle que soit la charge du système. Cependant le débit des liens, 10 Mo / sec, reste assez faible. Le C104 supporte deux techniques de routage : le routage explicite où l'émetteur du paquet inscrit dans l'en-tête du message la liste des ports de sortie successifs de chaque routeur à traverser, et le routage par intervalles, où les en-têtes contiennent un numéro absolu désignant le destinataire, et où la décision de routage est prise localement par chaque routeur en utilisant des tables de taille réduite. Le routage par intervalles présente de nombreux avantages, en particulier son traitement par des opérateurs matériels est relativement simple. Le C104 possède un mécanisme d'adaptabilité restreint, utilisable uniquement pour « grouper » des liens reliant deux même routeurs ensemble, et permettre le routage concurrent de plusieurs paquets. Cela est-il suffisant, en particulier pour bénéficier des topologies régulières possédant plusieurs chemins de taille équivalente ?

2-4 IBM SP2

Le réseau de l'IBM SP2 a été conçu avec une approche similaire à la nôtre. L'utilisation d'une interface découplée de la cellule processeur, pour permettre une évolution rapide des cellules processeurs, l'utilisation d'un réseau de communication point-à-point et d'un routeur de paquets sont les principaux points communs.

2-4-1 Architecture.

Le travail d'IBM a, tout comme le nôtre, porté sur la conception d'un système d'interconnexion rapide, dont la forte bande passante et la faible latence sont les enjeux principaux. Le SP2 possède une architecture à passage de messages. Une bibliothèque de fonctions logicielles est disponible pour les applications qui nécessitent un maximum de performances. Un portage optimisé de l'environnement de programmation PVM a été réalisé. Les entrées sorties peuvent être réparties sur les nœuds de calcul du système, contrairement au CRAY T3D.

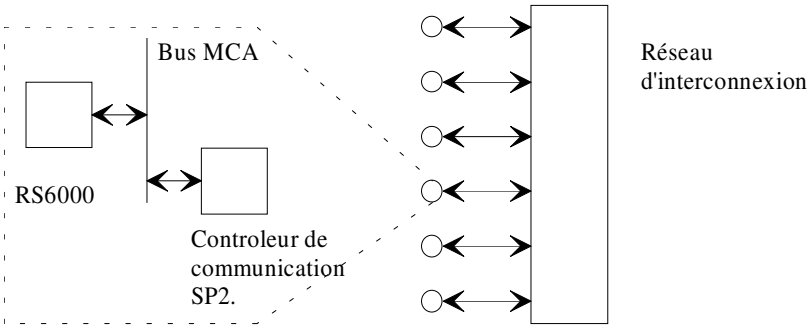

Figure 2-r : architecture du SP2.

Les nœuds de calcul sont composés d'un processeur RS6000, un contrôleur de cache secondaire et un bus MCA. L'un des périphériques présents sur le bus MCA est un contrôleur

- Il est simple à réaliser dans un composant. La taille des tables de routage est proportionnelle au nombre de sorties du routeur, et non au nombre de nœuds dans le réseau.
- Les opérations à effectuer dans chaque routeur sont simples et facilement parallélisables, la latence est donc réduite.

La figure 2-p présente un exemple de réseau C104 - T9000 et ses tables de routage par intervalles.

2-3-4 Adaptabilité par groupes.

Le C104 propose une méthode pour résoudre les problèmes de « points chauds », ou points de congestion.

Dans certains cas de routage, une liaison entre deux C104 va être sollicitée par plusieurs paquets simultanément. Ils sont alors routés de manière séquentielle, les uns après les autres. Le routage « wormhole » supporte mal ce genre de situation, car la contention va bloquer la queue des paquets, qui se trouve dans d'autres routeurs en amont sur le chemin de routage. Les contentions se rétro-propagent, car la queue de ces paquets va elle-même empêcher le routage d'autres paquets.

Pour éviter ce phénomène, le C104 possède une fonction dite de « Grouping », ou adaptabilité par groupes. Au lieu d'associer à un intervalle un unique port de sortie, on lui en associe un groupe. Ainsi les paquets possédant des en-têtes dans le même intervalle peuvent être routés simultanément par des liens physiques différents, mais équivalents.

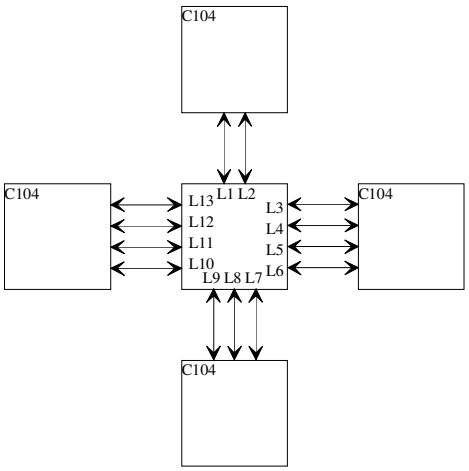

Figure 2-q : routage par groupes.

Deux limitations sont cependant présentes :

- Un port ne peut appartenir qu'à un seul groupe, et ce groupe ne contient que des ports adjacents.
- Il est impossible de définir une priorité entre ces liens; un paquet avec un en-tête dans l'intervalle associé au groupe sort donc par l'un des ports quelconques du groupe, sans que l'on puisse en définir un prioritaire.

Le routage par groupe ne peut être utilisé que dans un seul contexte : doubler les liens entre deux routeurs qui sont surchargés. On aurait préféré un mécanisme adaptatif plus complet, permettant aussi, dans les topologies où plusieurs chemins de taille identique existent, que les

Un autre avantage de ce dispositif est de fournir au récepteur un paquet directement en forme, débarrassé des octets qui n'étaient nécessaires qu'au routage.

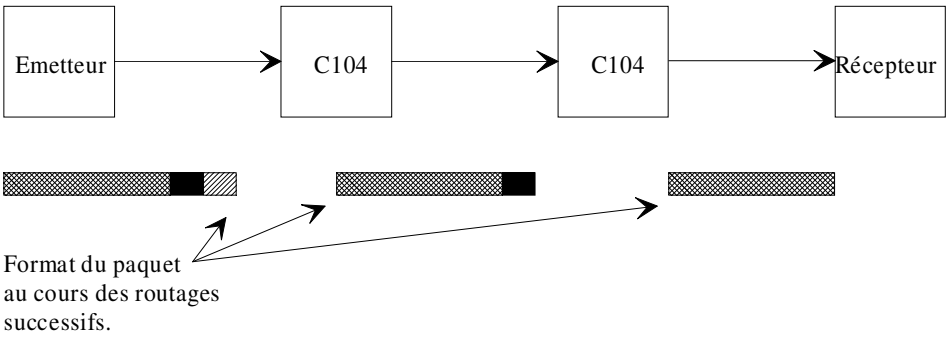

figure 2-o : routage explicite.

Le C104 propose néanmoins une autre technique de routage, permettant la réduction de la taille des en-têtes dans les topologies avec un grand nombre de nœuds, évitant à l'émetteur de calculer un chemin explicite. Il permet aussi la mise en œuvre de techniques de répartition automatique de charge (adaptative). Il s'agit du routage par intervalles.

Dans un premier temps, on affecte à chaque nœud du réseau un numéro unique. Ensuite on associe, dans chaque routeur C104, à chaque port de sortie, un intervalle dans l'espace des numéros des nœuds. La manière de numéroter les nœuds, ainsi que la construction des intervalles, peuvent être approfondies dans la bibliographie[22] plus en détail. Lorsqu'un routeur reçoit un paquet, il détermine à quel intervalle appartient l'en-tête du paquet, puis route celui-ci vers la sortie associée.

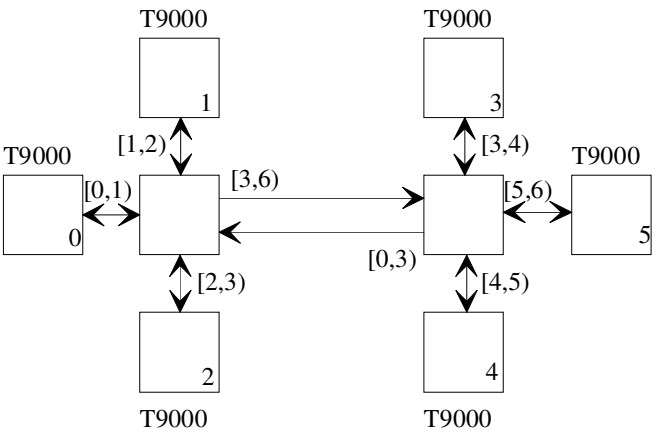

Figure 2-p : routage par intervalles.

Ce mécanisme présente quatre avantages principaux :

- N'importe quelle topologie est supportée par le routage par intervalles[23] .
- Le calcul des en-têtes par l'émetteur est simple, car il ne contient qu'une adresse absolue du destinataire.

[22] [LEEUWEN87] et [LEEUWEN91]
[23] [LEEUWEN87], [LEEUWEN91], [ZERR93] et [ZERR94]

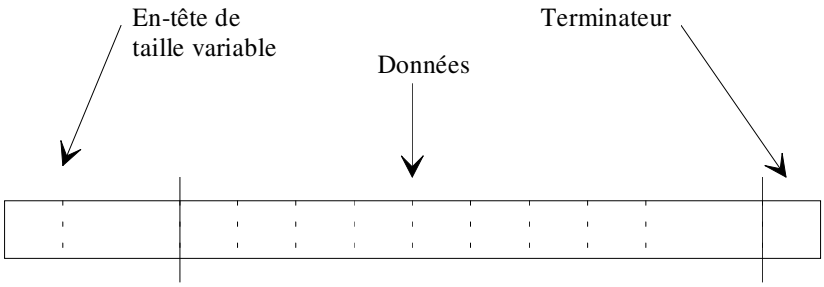

Figure 2-m : format des paquets.

2-3-2 Routage wormhole.

Le C104 est un routeur à 32 ports série. Il contient un crossbar complet, et peut donc router simultanément 32 paquets. Le routage utilise une technique dite « wormhole », c'est-à-dire que les paquets sont routés vers le port de sortie dès que leur en-tête a été analysé, avant même la réception de la totalité du paquet, afin de minimiser la latence.

Le lien utilisé pour le transport du paquet est libéré dès la transmission du caractère «End of Packet ». Ainsi les ressources utilisées pour la transmission d'un paquet sont réduites. La figure 2-n montre un cas de routage wormhole où deux liens, au maximum, sont utilisés simultanément pour le routage d'un paquet, libérant ainsi les autres pour le routage d'autres paquets.

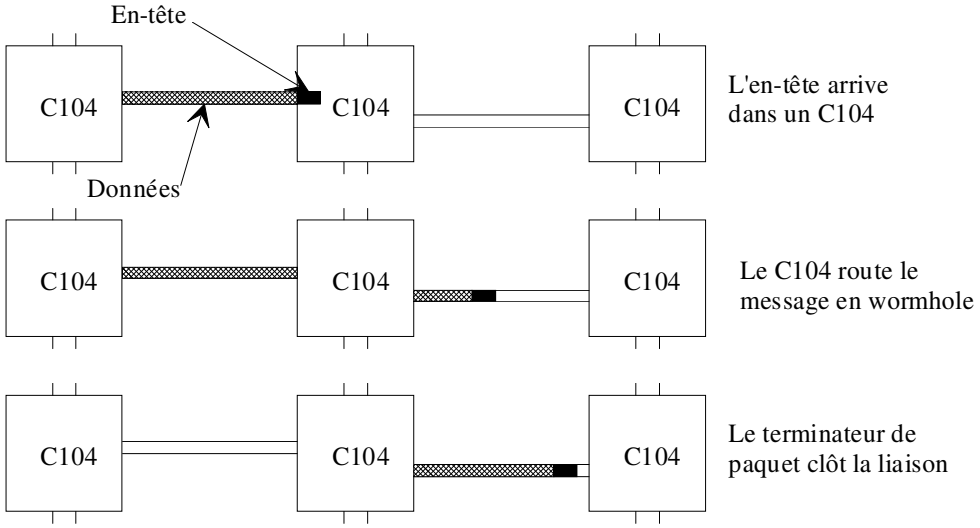

Figure 2-n : routage Wormhole.

2-3-3 Routage explicite et par intervalles.

Le C104 permet deux types de routage, le routage explicite, utilisant la technique de « Header Deletion », et le routage par intervalles.

Dans le routage explicite, l'en-tête est constitué par une suite de caractères, désignant le port de sortie à utiliser dans chaque routeur qu'il va traverser. Avec ce mécanisme, le chemin est calculé par l'émetteur du paquet qui met en forme son en-tête. Chaque routeur détruit le premier caractère du paquet. Ainsi, la décision de routage est simple : le paquet est envoyé vers le port de sortie désigné par le premier caractère du paquet.

2-3-1 DS-Link.

Le routeur C104 utilise, comme canal de communication, le lien série point-à-point bidirectionnel DS-Link. Ce lien transporte deux types de mots. Les caractères de données, qui contiennent 10 bits, et ceux de contrôle, qui contiennent 4 bits. La figure 2-l en donne les formats.

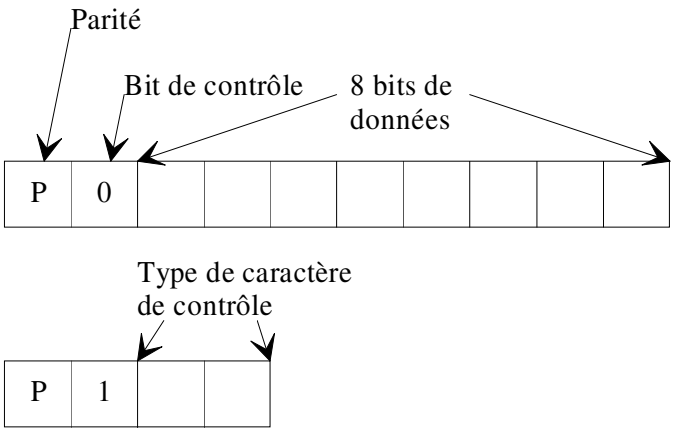

Figure 2-l : mot du DS-Link.

Il existe quatre caractères de contrôle :

- « Flow Control » : ce caractère spécial permet de régler le contrôle de flux. A chaque fois qu'un port récepteur du C104 a traité un ensemble de 32 caractères (un « flit »), il émet en retour un caractère de contrôle de flux autorisant l'émission d'un nouveau « flit ». Tout comme dans le réseau Myrinet, ce mécanisme est mis en œuvre pour éviter le débordement des mémoires tampon.
- « End of Packet » : ce caractère marque la fin d'un paquet. On peut voir le format des paquets sur la figure 2-m.
- « End of Message » : ce caractère marque la fin du dernier paquet d'un message.
- « Null » : Lorsque aucune transmission n'est en cours, le routeur C104 émet des caractères « Null » en permanence, pour permettre le bon fonctionnement des désérialiseurs du port récepteur auquel il est connecté.

Les paquets ainsi formés sont de taille quelconque. Le fait qu'il y ait deux terminateurs permet de détecter la réception complète d'un message sans avoir à garder une trace de tous les paquets reçus, en comptant les octets par exemple. Ce mécanisme impose cependant que les paquets d'un même message ne soient pas déséquentialisés dans le réseau.

- L'utilisation de pages mémoire préalablement vérouillées par le système d'exploitation pour permettre un dépot direct des messages dans la mémoire est une bonne solution, dans la mesure où elle limite « l'overhead » de réception.

On peut aussi poser plusieurs questions :

- Pourquoi utiliser une mémoire tampon dans le composant d'interface ? Ne peut-on imaginer un système où les données seraient copiées directement depuis la mémoire utilisateur vers le réseau, sans passer par une mémorisation dans l'interface ?
- Ne peut-on tenter d'implémenter le protocole TCP / UDP en évitant aussi les recopies en mémoire ? Il semble que cela soit possible, bien que les développements logiciels soient importants.

On reste aussi curieux du fonctionnement de l'API Myrinet. Comment le système gère-t-il la traduction entre la mémoire virtuelle et la mémoire physique ? Comment les différents processus se partagent-ils la mémoire gérée par l'API elle-même ?

Myrinet est un système de communication très performant. Les résultats, aussi bien en bande passante, qu'en latence, sont très bons. D'autre part, la disponibilité de ce matériel, pour les bus SUN et PCI, permet déjà une utilisation dans un large contexte, en particulier comme réseau local.

2-3 INMOS T9000 C104

La société INMOS propose un système de communication autour de deux composants le processeur T9000 et le composant de routage de paquets C104.

Le T9000[19] est un processeur qui intègre directement le matériel destiné à la communication (ou « transputers »). INMOS est spécialisé dans ce type de systèmes (voir aussi la J-Machine[20] du MIT). Cette famille de composants ne rentre clairement pas dans le cadre de notre problématique. Le processeur possède, en plus du contrôleur de communication, un « séquenceur » intégré dans le matériel. L'interaction entre ces deux sous-systèmes est très forte. Le jeu d'instructions « machine » est enrichi de deux primitives spécifiques « output message » et « input message » pour l'envoi et la réception de messages.

L'utilisation par un programme de ces instructions provoque un envoi ou une réception de messages, ainsi que le changement de contexte du processeur. Ceci permet de superposer efficacement les communications et le calcul. L'utilisation de canaux virtuels permet à plusieurs processus de s'échanger des messages simultanément, ceux-ci étant découpés en paquets de taille fixe.

L'utilisation de ce mécanisme, par ailleurs très performant, nécessite entre autres un système d'exploitation dédié. Elle présente les inconvénients liés aux solutions « non standard » tels que nous les avons décrites dans la problématique, c'est-à-dire la création de matériels et de logiciels spécifiques très complexes.

Par contre, le composant de routage INMOS C104[21] , routeur de paquets générique permettant de construire des réseaux à topologie quelconque, propose des réponses très pertinentes aux problèmes qui nous sont posés.

---

19 [THOMPSON]
20 [DALLY92]
21 [INMOS]

- Avec l'utilisation d'une bibliothèque de programmation (API) spécifique Myrinet, il est possible de copier directement les données depuis la mémoire de l'utilisateur vers celle de l'interface. Pour cela, la librairie d'interface Myrinet spécifique maintient un certain nombre de pages mémoire, déclarées comme « non-swapable » au système d'exploitation, qui permettent d'effectuer ces transferts.

Ce troisième mécanisme permet d'avoir des performances supérieures aux deux autres. Charles Seitz donne les résultats suivants, obtenus avec un bus SBUS à 20 Mhz :

Débit du lien Myrinet	80 Mo / sec
Débit TCP / IP Myrinet	9 Mo / sec
Débit API Myrinet	31 Mo / sec pour des paquets de 8 KOctets.
Latence matérielle	20 µs (déplacement de 32 octets)

La taille des messages permettant d'obtenir la moitié du débit maximum est de 2 KOctets, ce qui peut sembler un peu élevé pour des applications mettant en œuvre des transferts de messages de petite taille.

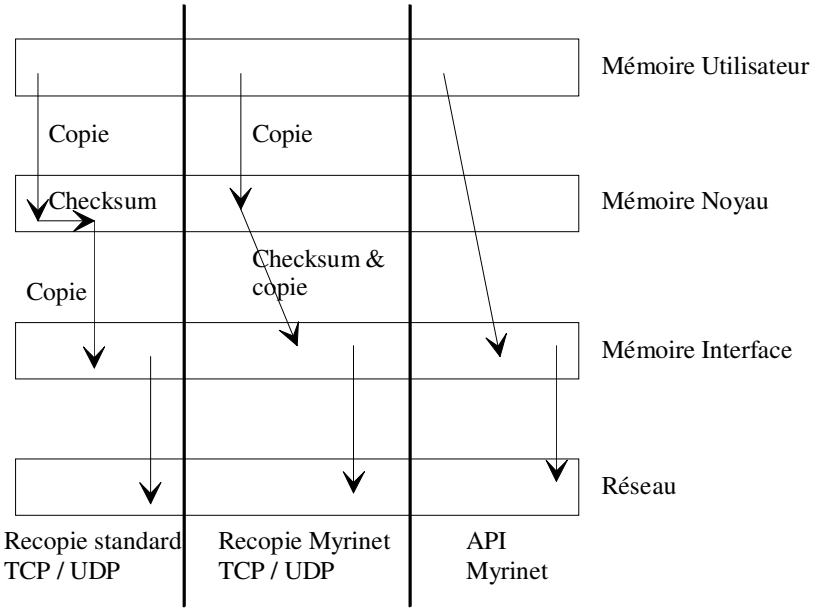

Figure 2-k : recopies mémoire dans les transferts TCP et UDP.

2-2-4 En résumé.

Ces résultats sont fort intéressants, on peut en retenir plusieurs points importants :

- Il faut éviter les recopies inutiles de données. Déplacer les données depuis la mémoire utilisateur vers le réseau est une bonne solution.
- Pour cela, l'utilisation de DMA est aussi une bonne solution, car elle décharge le processeur de cette tâche.
- Les performances du réseau Myrinet sont extrêmement bonnes. Le taux de transfert est particulièrement élevé.

2-2-3 L'interface LANAI.

L'interface LANAI, qui permet d'interconnecter une station de travail et le réseau d'interconnexion comprend 4 éléments distincts.

- Un composant VLSI spécifique, LANAI, contrôle le protocole des liens. Il comprend un processeur pour traiter les paquets, un contrôleur de DMA pour transférer les données depuis, et vers, la mémoire principale de l'hôte, et une interface vers les liens Myrinet.
- De la logique externe est nécessaire pour faire fonctionner le composant LANAI sur un bus de station de travail. Celle-ci est spécifique à chaque type de bus. Elle existe pour le moment pour le SBUS des stations SUN et le bus PCI.
- Des adaptateurs spécifiques permettent de faire l'adaptation entre le composant LANAI et les lignes différentielles des liens Myrinet.
- Une RAM de 32 K mots de 32 bits contient le programme exécuté par le processeur du LANAI et une zone pour mémoriser les paquets.

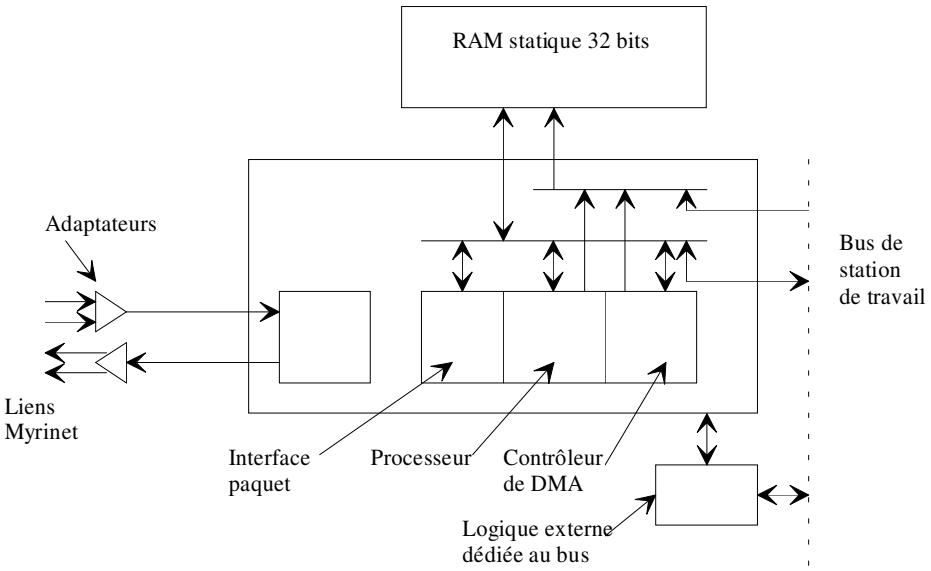

Figure 2-j : l'interface LANAI.

Le processeur interne du composant LANAI exécute un code spécifique, chargé au démarrage de la machine. Il interprète les paquets arrivant depuis le réseau, et génère les paquets à émettre. Pour cela, il dispose d'un contrôleur de DMA qui permet le transfert rapide des données avec la mémoire principale de la machine hôte.

Pour le moment, le système Myrinet est utilisé principalement pour encapsuler des trafics TCP et UDP. L'utilisation de liaisons point-à-point permet de s'affranchir des phénomènes de collisions des supports Ethernet, et donc d'obtenir de meilleures performances.

La figure 2-k présente les transferts mémoire habituellement présents dans les transferts TCP et UDP.

- Les mécanismes de TCP / UDP sont représentés à gauche. Deux copies successives des données sont effectuées, ainsi que le calcul des codes redondants.

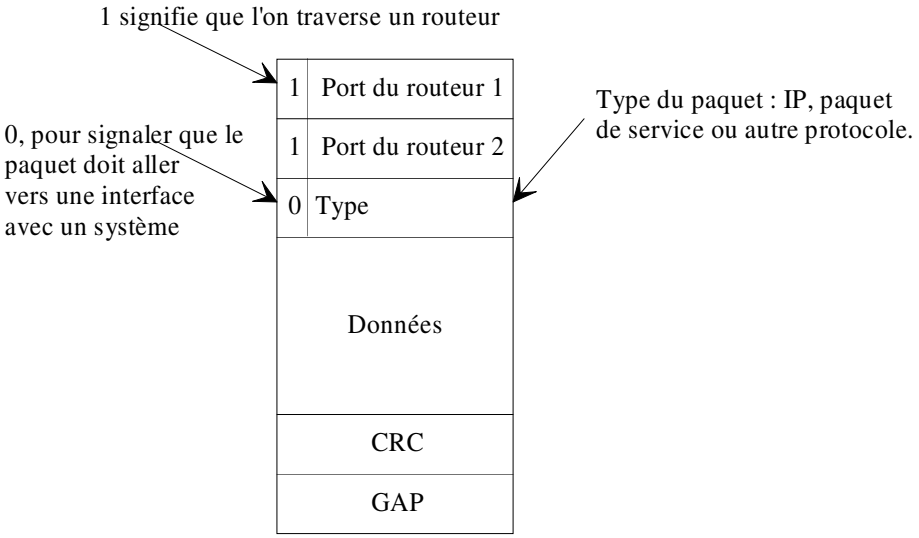

Figure 2-h : format des paquets Myrinet.

Ces routeurs utilisent une technique de routage « wormhole », c'est-à-dire que les paquets sont routés avant leur réception complète, après l'analyse de l'en-tête. Ceci permet de diminuer la latence globale du routage, et de gérer des paquets de taille quelconque. Dans un cas sans contention, la latence maximale de routage est de 550 nanosecondes pour un routeur à 8 liens.

La consommation de l'ensemble est de 15 Watts. Le routeur inclut deux composants électroniques spécifiques en technologie CMOS 0.8 µm.

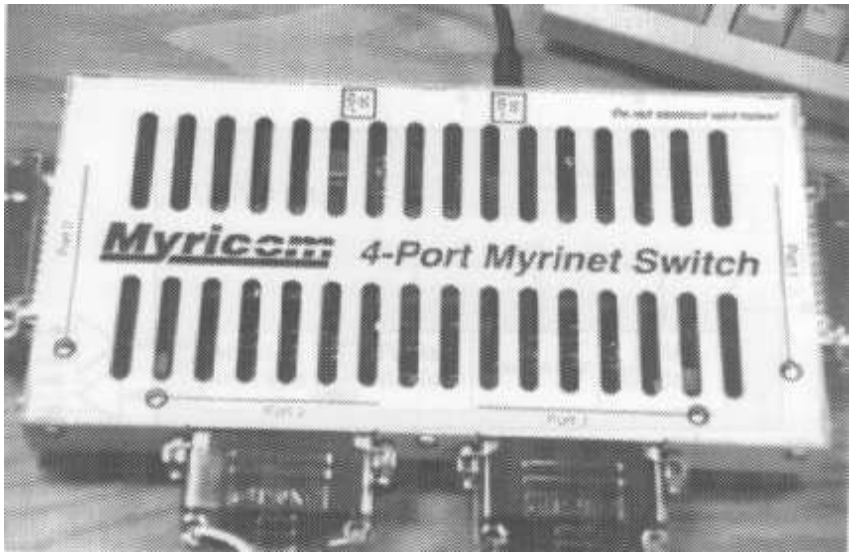

Figure 2-i : le routeur Myrinet.

Même si ses performances sont très bonnes, on peut regretter la taille importante de ce dispositif. Dans le cas de topologies avec un nombre important de machines, l'encombrement des circuits de routage est loin d'être négligeable. D'autre part, la connectique utilisée (18 paires torsadées par liaison) est, elle aussi, très encombrante. La longueur maximale des cables, 25 mètres, est un avantage de poids en faveur de ce dispositif.

mécanisme présente un hystérésis, dans la mesure où les seuils d'émission de « GO » et de « STOP » ne sont pas identiques, pour éviter que les oscillations autour d'un seuil unique entraînent la génération de multiples caractères de contrôle de flux.

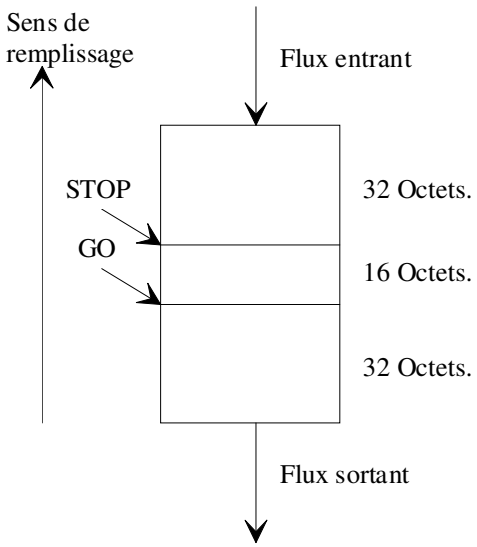

Figure 2-g : génération des caractères « STOP » et « GO ».

D'autre part, dans le cas où il n'y a pas de contention sur le réseau, aucun caractère de contrôle de flux ne circule, ce qui permet de ne pas perdre de bande passante. Par contre, dans les pires cas de contention, il se peut que deux caractères de contrôle circulent pour 16 caractères de données (soit 12 pour-cent de bande passante) ce qui encombre encore le réseau, juste au moment où l'on en a besoin.

2-2-2 Le routeur Myrinet.

Le routage utilisé par le système Myrinet est le routage explicite. Les paquets présentent le format présenté à la figure 2-h. L'en-tête du paquet est constitué par la liste des ports successifs des routeurs à traverser, suivie par un type de message, qui sert à la routine de réception, pour aiguiller le paquet vers les routines logicielles adéquates.

La taille des données contenue dans le paquet est quelconque. Un code cyclique permet une vérification de l'intégrité des données. Il est calculé sur la totalité du paquet, y compris l'en-tête.

Le routeur contient un crossbar complet, c'est-à-dire que, contrairement au CRAY T3D, des paquets peuvent circuler simultanément entre plusieurs entrées et plusieurs sorties, sans limitation due à une ressource interne.

Les routeurs Myrinet existent avec 4 ou 8 liens, des versions à 16 et 32 liens sont en cours de développement. Ce sont des cartes complexes, incluant de nombreux composants électroniques. Sur la figure 2-i, on peut voir une photographie de la version à 4 ports. Le clavier présent sur la photo permet d'avoir une idée de la taille du boîtier.

qualifient de « Message Passing Network »[16]. Ce dispositif permet de construire un réseau local pour stations de travail à faible latence et haut débit.

Il se compose de trois éléments principaux :

- Un lien de communication bidirectionnel parallèle sur 9 bits, muni d'un contrôle de flux.
- Un routeur de paquets à faible latence et haut débit. Ce routeur pilote quatre ou huit liens « full-duplex ». Des versions à 16 et 32 liens sont en préparation.
- Une interface entre un bus système et le lien de communication.

Ce système est l'héritier commercial de deux projets de recherches universitaires, Mosaic[17], de Caltech, et Atomic[18] de l'USC. Il est commercialisé par la société Myricom.

2-2-1 Le lien parallèle.

Le lien de communication du réseau Myrinet est un lien parallèle 9 bits. Il est composé de 18 paires torsadées, 9 dans chaque direction. Celles-ci font transiter des signaux différentiels avec une excursion entre 1 et 2 Volts.

Il s'agit d'une communication asynchrone dont le débit maximal est de 80 Méga-mots de 9 bits par seconde. Ces mots peuvent être ou bien des caractères de données, ou bien des caractères de contrôle spéciaux. Le 9 ^{ème} bit sert à les différencier. Le codage en transmission est un « NRZ », c'est-à-dire que l'on code les transitions sur les bits des caractères, 1 dans le cas d'une transition, 0 dans le cas d'absence de transition. Il existe un caractère « IDLE », composé uniquement de zéros qui est transmis en l'absence d'autres caractères à émettre.

Pour tolérer le « skew » (variation temporelle), et le « crosstalk » (couplage capacitif) entre les différents bits du mot transmis, la réception s'effectue de manière asynchrone, avec une circuiterie spécifique complexe.

Sur ce lien, on observe des taux d'erreurs inférieurs à 10^{-15}, si la longueur du câble reste inférieure à 25 mètres. Dans le cas de transmissions sur de plus longues distances, Myricom propose un adaptateur optique spécifique.

Le contrôle de flux est assuré par l'envoi en sens inverse du flux de données de deux caractères spéciaux « STOP » et « GO ». Ces caractères de contrôle, au 9 ^{ème} bit actif, sont multiplexés dans le flux de données et envoyés en sens inverse du flux dont on veut réguler le flot.

- Lorsqu'un routeur ne peut plus accepter de caractères, parce que ses mémoires tampon internes sont saturées, il envoie un caractère « STOP ».
- Lorsque la mémoire tampon se libère de nouveau, c'est-à-dire que le réseau a résorbé la contention, et que les paquets peuvent continuer leur cheminement, il envoie un caractère « GO ».

Comme l'aller-retour de ces caractères de signalisation met un certain temps, à cause essentiellement de la longueur du câble, et du temps nécessaire pour la génération et l'interprétation des caractères « STOP » et « GO », leur génération se fait de manière anticipée, comme on peut le voir sur la figure 2-g.

Le caractère « STOP » est émis lorsque la file d'attente de réception dépasse le seuil de 48 octets, le caractère « GO » est émis lorsque son niveau revient en dessous de 32 octets. Ce

16 [SEITZ95]
17 [SEITZ93]
18 [FELDER]

différentes. Or des opérateurs de resynchronisation, intégrés dans le silicium du réseau d'interconnexion, sont beaucoup moins coûteux que la distribution, dans un gros système, d'une horloge unique. Evidemment, les débits de transferts proposés par les CRAY (près de 4 Gbits / sec par liaison pour le CRAY T3E) ne sont pas encore réalisables sur un lien série asynchrone en technologie CMOS, et la distribution de cette horloge unique permet de proposer des débits très importants. Des recherches sont en cours, conduites par Roland Marbot chez SGS-Thomson, pour permettre la réalisation d'une liaison série asynchrone à 3 Gbits / sec, qui permettra donc d'obtenir des débits similaires à ceux des machines CRAY, sans pour autant devoir distribuer une horloge unique.

2-1-4 En résumé.

Nous avons succinctement présenté le système d'interconnexion et l'interface réseau du CRAY T3D.

Le réseau utilise des liaisons point-à-point parallèles et des routeurs de paquets. Ceux-ci sont dédiés à la topologie du CRAY T3D, c'est-à-dire le tore tridimensionnel, dans la mesure où ils implémentent l'algorithme de routage par dimension ordonnée. L'architecture du routeur ne présente pas de crossbar complet, ce qui interdit certains types de routages concurrents. L'utilisation de mémoires tampon multiples permet d'éviter les interblocages sur les sous-parties cycliques du réseau. Même si cette technique impose le doublement de la capacité de mémorisation dans tous les éléments de routage, elle est particulièrement efficace.

La communication entre les processeurs est de type partage de mémoire, utilisant des primitives de lecture et d'écriture à distance, agrémentées de mécanismes de synchronisation. Cependant, un mécanisme de type « Message-Passing » est implémenté. Celui-ci implique une forte participation du processeur émetteur, qui doit explicitement construire les sous-paquets du message et effectuer des retransmissions en cas d'erreur, et une forte participation du processeur récepteur, qui doit effectuer des traitements logiciels pour extraire les messages de la file d'attente où le système d'interconnexion les range.

Le protocole pour ces transferts de type « Message-Passing » utilise une file d'attente pour réceptionner les messages. Ceci implique la mise en œuvre d'un protocole d'acquittement, générant un message d'acquittement pour chaque message reçu, qui est ici géré par le matériel, mais qui affecte la bande passante utilisable du réseau. Six octets d'acquittement pour 32 octets de données, soit 18% de la bande passante, sont utilisés pour la transmission de ces acquittements.

La gestion des périphériques est effectuée, soit par l'ordinateur hôte du système, soit par des composants externes au T3D. Certains nœuds contiennent donc des liaisons rapides pour dialoguer avec d'autres systèmes informatiques.

Le système est entièrement synchrone. Une seule horloge est distribuée dans tout le système, ce qui implique un surcoût de construction de la machine, qui est peut-être négligeable devant les coûts respectifs des autres éléments, mais qui est incompatible avec notre approche.

2-2 Myrinet

Le réseau Myrinet est, de ceux que nous présentons ici, l'un de ceux qui répond le plus directement à notre problématique. Danny Cohen et Charles Seitz, les principaux architectes, le

Cependant, deux fonctions supplémentaires présentes dans le CRAY T3D nous intéressent plus directement. Elles mettent à la disposition des programmes un modèle par passage de messages, qui permet de porter plus rapidement des applications conçues sur d'autres machines.

Message-Passing.

Un processeur peut envoyer un message sur le réseau en direction d'un autre processeur.

Pour émettre le message, le microprocesseur émetteur construit des paquets de 4 mots de 64 bits qu'il transfère à la circuiterie spécifique en effectuant une écriture dans une zone de l'espace d'adressage. Les paquets sont aussitôt émis sur le réseau d'interconnexion.

Les paquets sont de taille fixe, ils comprennent 6 mots de 16 bits d'en-tête, et 4 mots de 64 bits de données, soit 44 octets.

A la réception d'un paquet, la circuiterie auxiliaire tente de le ranger dans une file d'attente. Si cette file d'attente présente des emplacements libres, une interruption est déclenchée sur le processeur récepteur, et un acquittement est émis vers le processeur émetteur. Ensuite, le logiciel vient consulter cette file d'attente, et range les données dans la mémoire.

Si cette file d'attente est pleine, un acquittement négatif est renvoyé vers le processeur émetteur pour que celui-ci réémette le paquet. On note que la taille de file d'attente (4 kilo-messages) permet d'estimer que cette condition n'apparaît pas souvent.

Ce mécanisme permet d'assurer que la transmission des paquets se fait « à coup sûr », dans la mesure où l'émetteur est prévenu que le paquet n'a pu être transmis au processeur récepteur. A la réception de l'acquittement négatif, le microprocesseur émetteur doit réémettre le message.

Cependant ce mécanisme présente, en terme de gestion de la bande passante du réseau, un inconvénient.

L'utilisation de files d'attente pour mémoriser les messages en réception implique la gestion d'acquittements. Ceux-ci consomment une partie de la bande passante. La taille limitée des données présentes dans chaque paquet (32 octets) rend le coût de ces acquittements non négligeable, même si leur taille est particulièrement réduite (6 octets).

2-1-3 Les ponts d'entrée-sortie et l'horloge centralisée.

Des ponts d'entrée-sortie sont présents dans le CRAY T3D. Il s'agit également de deux processeurs DEC Alpha avec la circuiterie de communication associée, à laquelle sont ajoutés deux liens haut débit. L'un permet la connexion avec la machine « hôte », l'autre vers un éventuel système d'entrée-sortie.

Ces liens servent à la configuration du T3D, au contrôle des applications qui s'y déroulent (démarrage, terminaison, récupération des résultats) et à permettre à celui-ci d'accéder aux périphériques : disques, lecteur de bande etc..

Le système d'horloge centralisé du CRAY T3D est certainement le point le plus discutable. Distribuer une horloge à 150 Mhz à tous les composants d'un système pouvant couvrir plusieurs mètres cube n'est pas une mince affaire. Evidemment, la supposition a priori du synchronisme des différents sous-systèmes permet de s'affranchir de nombreux problèmes électriques. Cependant, il existe aujourd'hui de nombreuses techniques qui permettent de resynchroniser, en des délais très faibles, des signaux numériques cadencés par des horloges

- Soit la zone de mémoire accédée correspond à des registres internes de ces composants spécifiques.

Les accès mémoires peuvent être de 3 types différents :

- Les opérations de lecture « non-cachables » : le microprocesseur lit une donnée, éventuellement détenue par une autre cellule processeur. La donnée est lue, mais n'est pas placée dans le cache du processeur. L'accès suivant à cette donnée distante impliquera donc obligatoirement la création de nouveaux trafics sur le réseau.
- Les opérations de lecture « cachables » : cet accès est identique au précédent, si ce n'est que les données sont inscrites dans le cache du processeur. Lors d'accès ultérieurs en lecture à cette donnée, aucun trafic sur le réseau ne sera nécessaire.
- Les opérations d'écriture : les opérations d'écriture sont soit dirigées par les composants spécifiques vers la mémoire locale, soit transformées en un message sur le réseau pour effectuer une écriture distante.

Diverses primitives de synchronisation du type « barrière de synchronisation », « Fetch and Add », complètent ces opérations essentielles, pour permettre le partage effectif de zone de données entre les processeurs. Une autre, « pré-chargement de données », permet d'anticiper l'acquisition de données qui seront ultérieurement nécessaires, et de paralléliser ainsi les communications et le calcul.

Ces opérations de gestion de mémoire partagée sont particulièrement intéressantes. Elles sont ici simples à mettre en œuvre, car le système ne possède pas de mémoire virtuelle, et ne gère pas le temps partagé. Elles permettent aux applications de voir de manière transparente la totalité de la mémoire du système, autorisant une synchronisation simple de celles-ci. La totalité des structures de données présentes dans chacune des cellules processeurs peut être vue par les autres, de manière presque transparente pour les logiciels. Les programmes partageant des données effectuent des accès non « cachables » sur celles-ci, et effectuent des accès « cachables » sur les données dont ils sont l'unique utilisateur.

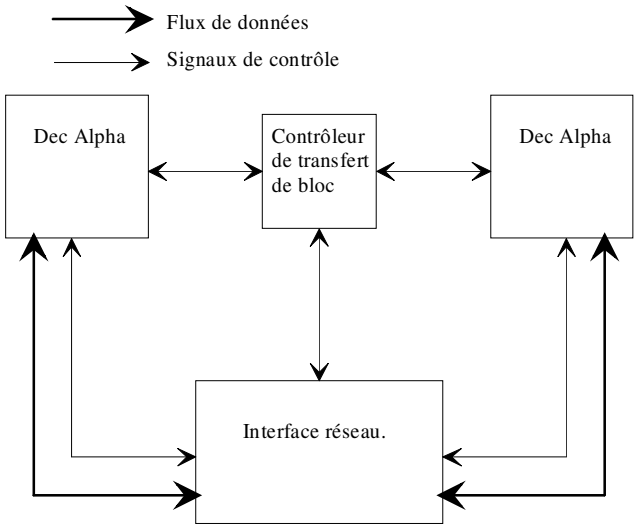

Figure 2-f : structure des nœuds du CRAY T3D.

Cette approche, nous l'avons vue dans la problématique, implique l'écriture de systèmes d'exploitation et d'applications spécifiques qui ne rentrent pas dans le cadre de notre étude.

On peut voir sur la figure 2-d que l'interblocage est évité. Au moins l'une des deux communications de 0 vers 2 et de 1 vers 3 peut s'effectuer. En effet, la première qui se présente sur le lien de 1 vers 2 se voit attribuer la mémoire tampon 1 du nœud 2. Le message peut donc transiter. Une fois que celui-ci est effectué, le routeur route successivement les autres messages.

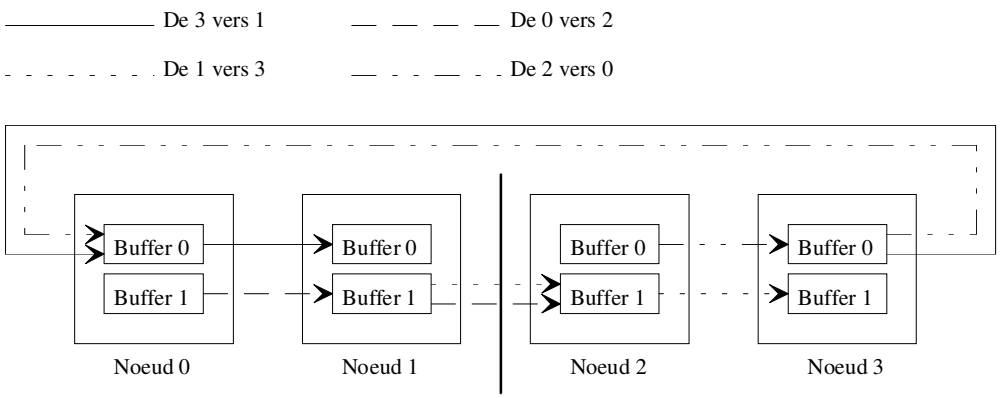

Figure 2-d : canaux virtuels de communication.

Ce système est relativement complexe. Il impose un doublement de la mémorisation dans tous les composants de routage. Une autre solution pour éviter les interblocages dans les cycles des réseaux à routage wormhole consisterait à dupliquer certains des liens entre deux des processeurs, comme on peut le voir sur la figure 2-e. Le mécanisme fondamental est le même : l'interblocage est évité et les messages sont routés successivement.

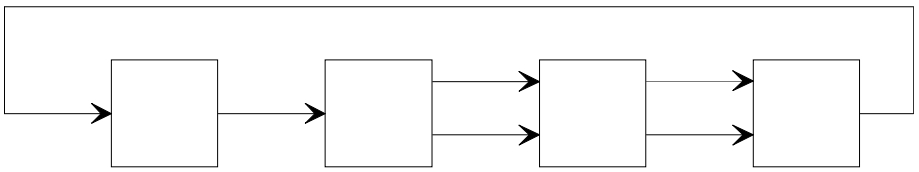

Figure 2-e : gestion de l'interblocage dans les réseaux à une seule mémoire tampon.

La solution proposée par CRAY est une excellente. Le doublement des éléments de mémorisation du routeur est moins coûteux que le doublement des liens de communication.

2-1-2 La cellule processeur.

Les nœuds du système sont des cartes bi-processeurs DEC Alpha à 150 Mhz. Deux composants spécifiques sont également présents. L'un permet d'effectuer des transferts de zones de données; il agit comme un contrôleur de DMA capable d'effectuer des recopies à distance. L'autre sert d'interface au réseau.

Le CRAY T3D est essentiellement une machine à mémoire partagée, c'est-à-dire que chaque cellule processeur contient une partie de la mémoire totale du système. Un accès d'un processeur à une adresse physique est interprété par les deux composants spécifiques.

- Soit l'accès correspond à de la mémoire locale, présente physiquement sur la cellule processeur. Le processeur accède alors directement à la donnée.
- Soit cet accès correspond à de la mémoire distante, il est alors transformé en accès réseau.

Les messages routés sont de taille limitée. L'en-tête d'un message contient la source et la destination du message. Il est suivi par des données, dont le nombre est spécifié dans l'en-tête. Des codes redondants permettent aux récepteurs de détecter les éventuelles erreurs qui se seraient produites sur les données pendant le routage.

Le routage effectué est un routage « wormhole », c'est-à-dire que le routage est effectué dès la réception de l'en-tête, avant que la totalité des données soit arrivée au routeur.

Les messages transportés sont de taille fixe, supérieure à la capacité des mémoires tampons de réception du routeur. En cas de contention, un message n'est donc pas entièrement contenu dans un élément de mémorisation du routeur, ce qui implique le blocage des routeurs précédents sur le chemin qu'il a parcouru, interdisant le routage concurrent d'autres messages.

La figure 2-c présente la structure interne du routeur. Il est composé de trois sous-éléments de routage, dédiés chacun à une direction. Cette structure ne permet pas des routages concurrents quelconques au sein d'un routeur, même si les messages transitent depuis et vers des liens différents. En effet, un message entrant par le lien +X et sortant par le lien -Z interdit le routage d'un message depuis -X vers +Z, à cause d'une contention sur une ressource interne, alors que le réseau devrait le permettre. CRAY n'insiste pas sur cette perte de bande passante globale du système due à l'architecture interne du routeur, mais il serait intéressant de la chiffrer précisément.

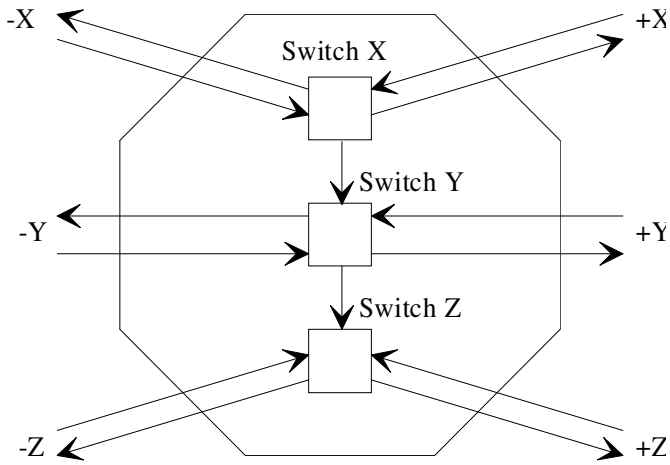

Figure 2-c : structure interne du routeur.

Pour l'algorithme de routage décrit ci-dessus, le routage wormhole peut présenter des conditions d'interblocage.

Afin de remédier aux éventuels interblocages, le routeur possède deux mémoires tampon, qui permettent aux paquets de suivre des canaux virtuels différents. On peut en voir une illustration sur la figure 2-d.

Sur celle-ci, on peut voir une condition d'interblocage évitée grâce aux canaux virtuels. Simultanément, chacun des processeurs, sur une même direction X, envoie un message à un processeur distant de deux liens. Le logiciel choisit un lien au démarrage du système, ici le lien reliant 1 à 2, pour casser les interblocages.

Si une communication entre deux processeurs utilise ce lien, elle transite toujours par les mémoires tampon 1. Si une communication n'utilise pas ce lien, elle transite toujours par les mémoires tampon 0.

Le réseau d'interconnexion permet l'échange de données entre les différents éléments de calcul. C'est un tore tridimensionnel, représenté sur la figure 2-b. Le routage des messages depuis un nœud A vers un nœud B s'effectue par « dimension ordonnée ». C'est-à-dire que le message traverse d'abord le réseau depuis A suivant la direction X, jusqu'à atteindre le plan contenant le nœud B. Il continue ensuite, suivant la dimension Y jusqu'à la droite contenant B, puis suivant la direction Z vers B.

Le tore tridimensionnel est une matrice tridimensionnelle dont les nœuds à la périphérie sont reliés entre eux, comme on peut le voir sur la figure 2-b.

Schématiquement, le routeur route le message sur le lien qui lui permet de se rapprocher du plan (x), puis de la droite (x, y), et enfin du nœud (x, y, z) (figure 2-a).

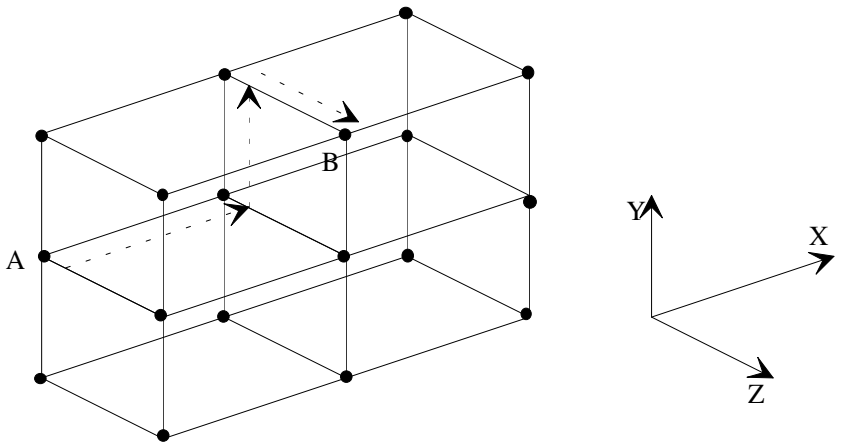

Figure 2-a : matrice tridimensionnelle.

Cette topologie étendue permet de résister aux pannes de certains liens. Sur chaque dimension, tout comme sur un anneau, un lien en panne peut-être contourné en effectuant le routage dans la direction opposée, tout le long de l'anneau (figure 2-b). La connexion des nœuds situés aux extrémités permet aussi de diminuer le diamètre du réseau par deux.

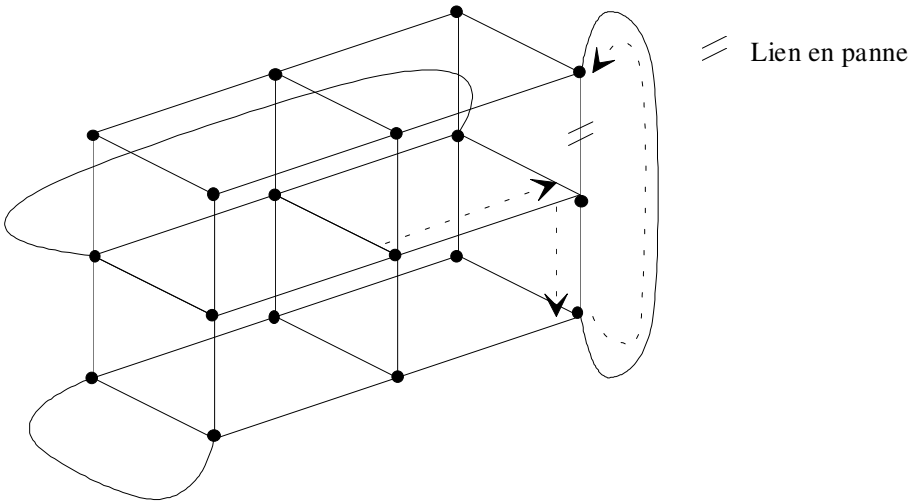

Figure 2-b : tore à trois dimensions.

Le routeur du T3D est dédié à cette topologie, dans la mesure où il effectue l'algorithme de routage par dimensions successives que nous avons évoqué.

Chapitre 2 Etat de l'art.

On peut diviser les réseaux d'interconnexion des machines parallèles en deux familles.

Tout d'abord, ceux qui présentent une approche similaire à la nôtre, c'est-à-dire les réseaux dont l'interface avec la cellule processeur est placée sur un bus standard. C'est en particulier le cas du réseau Myrinet et de l'IBM SP2. Ces deux réseaux proposent des réponses particulièrement intéressantes et originales aux problèmes soulevés précédemment.

D'autres projets de réseaux d'interconnexion proposent des approches différentes de la nôtre. Dans le CRAY T3D ou les systèmes à transputer INMOS C104 - T9000, le couplage entre le processeur et l'interface réseau est bien plus fort. Dans le CRAY, l'interface réseau est intégrée au gestionnaire de mémoire. Dans les systèmes à transputer, elle est incluse dans le processeur lui-même. Cependant, un grand nombre de caractéristiques de ces systèmes ont été utilisées dans le nôtre. Il s'agit donc de les présenter succinctement et de montrer en quoi ils sont particulièrement intéressants.

2-1 Le CRAY T3D.

Le CRAY T3D[15] est un ordinateur parallèle, contenant jusqu'à plusieurs milliers de processeurs. C'est un système non autonome, qui se connecte à un ordinateur « hôte ». Ce système hôte assure la compilation des programmes et gère l'exécution de ceux-ci sur la machine parallèle. La machine « hôte » exécute en particulier le système d'exploitation et gère les périphériques d'entrée-sortie.

Quatre types d'entités sont présentes dans un CRAY T3D.

- Les processeurs élémentaires : ce sont des microprocesseurs 64 bits DEC Alpha 21064. Ils possèdent, chacun, entre 16 MOctets et 64 MOctets de mémoire. CRAY propose aujourd'hui le CRAY T3E, dont l'architecture est identique, mais qui possède des processeurs 21164 plus puissants.
- Le réseau d'interconnexion : c'est ici un tore tridimensionnel. Il est composé de liaisons point-à-point et de routeurs de messages.
- Les ponts d'entrée-sortie : ceux-ci permettent le transfert de données entre le système hôte et le T3D, ou bien entre celui-ci et des périphériques d'entrée-sortie.
- Une horloge centrale : une horloge est distribuée à tout le système. Tous les éléments y fonctionnent de manière totalement synchrone, à 150 Mhz.

Le CRAY T3D est une machine dédiée au calcul scientifique. Les processeurs ne possèdent qu'un système d'exploitation réduit, ne gérant en particulier pas le temps partagé, et n'incluant pas de gestionnaire de mémoire virtuelle. Toutes les adresses que nous mentionnons dans la suite de cette partie sont donc des adresses physiques.

2-1-1 Le réseau d'interconnexion.

Les liaisons point-à-point du réseau sont des liens parallèles sur 16 bits. Ici, à chaque cycle, 16 bits du même mot sont transportés.

---

15 [T3D]

Problématique.

1-3 Conclusion.

1-3-1 Les performances.

L'enjeu est certainement celui des performances. Jusqu'à quel point peut-on reporter les performances du lien HSL en terme de débit et de latence ? Notre réponse à cette question se limite à une évaluation précise, par simulation, de l'impact des composants et du protocole de communication de notre réseau sur le débit et la latence.

1-3-2 Le coût.

L'un des autres enjeux de notre étude est de maîtriser les coûts, aussi bien de développement que de production. Peut-on trouver un bon compromis coûts de production / complexité / performances ?

Voici donc deux questions auxquelles nous nous efforcerons d'apporter des réponses positives.

données mises en œuvre pour déterminer le cheminement d'un message depuis un processeur vers un autre reste raisonnable. Il faut que la latence augmente moins vite que le nombre de processeurs, pour pouvoir garantir un « speedup » raisonnable aux machines avec plusieurs centaines de processeurs.

Il faut aussi que le système soit économiquement viable avec un petit nombre de processeurs.

1-2-4 Overhead.

Un autre point important dans la conception des réseaux d'interconnexion de machines parallèles est « l'Overhead » : il faut que la puissance de calcul consommée par les opérations d'émission et de réception de données soit minimale. Le composant d'interface au réseau devra donc présenter des mécanismes qui permettent de soulager au maximum le processeur du traitement des messages entrants et sortants. Ces aspects sont étudiés en détail dans un autre article de David Patterson[12] .

1-2-5 Le coût.

Le coût constitue sans doute l'un des points les plus importants de notre démarche. Nous voulons minimiser les coûts de développement et de fabrication dans tous les éléments de notre réseau. Réduire les coûts veut essentiellement dire « faire simple », sans pour autant sacrifier les performances. On peut énoncer deux principes qu'il faudra systématiquement appliquer :

« Hardware should provide primitives - not solutions »[13] . En particulier, les composants doivent permettre des utilisations logicielles variées. Modifier du logiciel est beaucoup moins coûteux que modifier un composant électronique.

« Support the most frequent case in hardware »[14] . Il faut que les composants matériels exécutent les opérations les plus fréquentes le plus rapidement possible. Les opérations rares, ou exceptionnelles doivent être effectuées en logiciel. C'est aussi le principe des processeurs RISC.

Plus que le prix au sens économique du terme, c'est la volonté d'achever nos réalisations qui nous guide. C'est-à-dire que la faisabilité de chaque mécanisme, de chaque opérateur, doit être confrontée aux moyens matériels et humains de notre équipe.

1-2-6 La pérennité.

Enfin, le réseau d'interconnexion doit pouvoir supporter plusieurs cycles de vie de processeurs. En effet, les temps de développement pour notre projet sont non nuls, et il serait dommage qu'une fois la conception terminée, le système ait déjà deux ans de retard.

De la même manière que l'utilisation de cartes mères de stations de travail comme cellule monoprocesseur, ou l'utilisation de systèmes d'exploitation standards permet d'intégrer rapidement les dernières technologies, notre système doit pouvoir intégrer sans modification, ou rapidement, les évolutions de la technologie.

Nous avons présenté nos objectifs. Evidemment, nous serons amenés à faire des compromis pour pouvoir les réaliser, mais nous pouvons déjà résumer les enjeux de cette étude.

---

12 [PATTER95]
13 [HEPA]
14 [KATE]

- La liaison série nous permet de minimiser les coûts du système. En effet, un simple câble coaxial permet de transporter les informations, ce qui est beaucoup moins coûteux que le routage d'un bus sur une carte d'extension.
- L'utilisation de la technologie CMOS va nous permettre de développer des composants à faible coût de production.
- Le bloc qui permet la transmission et la réception d'un octet présente une surface très faible, de l'ordre de 1,5 mm^2, et sa consommation est par ailleurs réduite (0,3 Watt).

Cette étude vise donc la construction d'un réseau d'interconnexion et son interface avec les stations de travail du marché permettant la construction de machines parallèles.

Le cadre de notre étude est désormais fixé. Nous allons maintenant présenter nos objectifs précis.

1-2 Les objectifs.

1-2-1 Le débit.

Tout naturellement, le débit du réseau d'interconnexion est un point critique. Nous savons que les liens série sont susceptibles de transférer 1 Gbit / sec. Cependant il faudra s'assurer que l'ensemble de notre réseau peut soutenir ce débit, c'est-à-dire l'interface au processeur et les composants de communication.

Tout naturellement, il faut que ce débit soit utilisable par l'application. Il faudra donc s'employer à proposer une solution permettant de fournir à l'application une grande partie du 1 Gbit / sec.

1-2-2 La latence.

La latence est, avec le débit, l'élément le plus important de notre système. La latence d'un transfert est la durée écoulée entre le début de l'émission, et la fin de la réception.

Un exercice particulièrement difficile est demandé au programmeur dans les systèmes à forte latence : pour avoir de bonnes performances, il doit anticiper les communications, en tenant compte de leur latence, et leur superposer des calculs. Dans certains cas, ceci n'est pas possible, c'est-à-dire que la puissance de calcul est perdue à attendre que le réseau d'interconnexion achemine les données[11] .

Il faut que la latence soit faible, quelle que soit la taille des unités d'informations échangées et même en cas de trafic très intense sur le réseau.

Là encore, la difficulté consiste à effectivement reporter cette faible latence jusqu'à l'application par l'utilisation de méthodes logicielles adéquates.

1-2-3 L'extensibilité.

Il faut que nous puissions intégrer un très grand nombre de nœuds (plusieurs centaines) à notre réseau d'interconnexion, tout en gardant des performances acceptables. L'utilisation de liaisons point-à-point rentre évidemment dans ce cadre, mais il faut pouvoir concevoir des réseaux avec des topologies quelconques, éventuellement non régulières, où la taille des structures de

[11] [CYPHER]

bénéficier de la performance du système d'interconnexion. Une autre solution est de porter l'interface des réseaux locaux : le modèle des sockets. Cette solution permet d'utiliser les outils des réseaux locaux, en supprimant le protocole TCP / IP.

1-1-7 Le médium : le lien de communication HSL.

Comme nous l'avons vu, pour transformer cette collection de cellules monoprocesseurs en machine parallèle, il faut disposer d'un mécanisme de communication. Ce mécanisme doit permettre aux processeurs d'émettre et de recevoir des données avec des débits importants.

D'autre part, le phénomène de standardisation évoqué ci-dessus ne s'est pas encore étendu aux réseaux d'interconnexion. Certes, ATM ou SCI proposent actuellement une réponse aux contraintes que posent les machines parallèles à leur réseau d'interconnexion, mais leur domination sur le marché n'est pas encore assurée. D'autre part, leurs caractéristiques ne les rendent pas forcément tout à fait adaptés à la construction de machines parallèles.

ATM est un réseau d'interconnexion à commutation de paquets. Il a été conçu pour répondre aux besoins des transferts de données à haut débit sur de longues distances. C'est donc avant tout un protocole de télécommunication. Intrinsèquement, ATM possède un niveau de complexité inutilement élevé pour une utilisation dans une machine parallèle. Les contraintes du protocole dues à l'utilisation d'ATM pour des transferts sur de très longues distances (quelques milliers de kilomètres) introduisent dans les éléments de commutation des mécanismes complexes. Ces mécanismes sont inutiles dans le cadre d'une utilisation pour une machine parallèle, et introduisent une latence qui sera toujours supérieure à celle d'équipements spécifiquement construits pour des applications locales.

La technologie SCI (IEEE 1596) est un réseau local haute performance. Elle impose au réseau une topologie en anneau, même si aujourd'hui des ponts peuvent être placés entre différents anneaux. Un autre inconvénient majeur de la technologie SCI, est qu'elle requiert pour le moment des composants en technologie ECL ou AsGa.

Les standards actuels ou émergents ne remplissent pas toutes les contraintes que nous nous sommes fixées. Notre démarche se situe dans ce cadre, en coopération avec de nombreux industriels tels que BULL ou SGS-Thomson, autour de la technologie HSL. Il s'agit, pour notre équipe, de proposer un réseau d'interconnexion permettant la construction de machines parallèles dans des contextes d'utilisation très divers.

HSL.

Le lien HSL est un lien série point-à-point à 1 Gbit / seconde. Une macro-cellule réalisant les fonctions de sérialiseur et de désérialiseur en technologie CMOS est disponible[10]. Cette technologie, initialement développée par la société Bull et le laboratoire MASI est aujourd'hui intégrée au standard IEEE 1355.

- 1 Gbit / seconde correspond ici, compte tenu du codage d'un octet sur 12 bits, à 80 MOctets / seconde.
- La liaison point-à-point nous permet de créer des topologies avec un grand nombre de processeurs (quelques dizaines, voire centaines). Au-delà d'une vingtaine de processeurs, les solutions utilisant des bus sont souvent peu performantes.

[10] [HSL]

l'application de la loi de Gordon Bell, qui indique que le doublement de la quantité de production d'un système électronique diminue son prix unitaire de 90%. On y voit clairement que la solution la moins coûteuse est d'employer 32 serveurs quadri-processeurs Sun Sparc 10.

L'idée d'employer une carte mère d'une station de travail du marché comme cellule monoprocesseur est donc économiquement pertinente. Elle reste pertinente au niveau des performances si elle permet à ces nœuds standards de communiquer efficacement. C'est possible aujourd'hui avec les bus d'entrée-sortie rapides, tel que le bus PCI, qui sont intégrés aux cartes mères du marché. Nous le voyons plus en détail dans le chapitre 4.

L'utilisation de cartes mères de station de travail comme nœuds de traitement se justifie tout autant dans les machines génériques que dans les machines dédiées, dans la mesure où les contraintes de coût sont du même ordre.

1-1-5 Le système d'exploitation : un Unix standard.

Le même type de raisonnement qui conduit à choisir une carte mère de station de travail comme cellule processeur, amène aujourd'hui à utiliser comme système d'exploitation un système standard.

Les mêmes avantages se présentent, le coût est faible, la certitude de bénéficier le plus rapidement des dernières avancées dans le domaine des systèmes d'exploitation est assurée, ainsi qu'une gamme large de pilotes logiciels pour les périphériques.

Quelques ajouts doivent être effectués sur ce système d'exploitation, de manière à intégrer les mécanismes de communication. Ceci implique aussi le choix du modèle de programmation par passage de message, qui est le seul à s'intégrer simplement et efficacement à Unix. Par ailleurs un grand nombre de logiciels ont déjà été conçus pour Unix, et leur portage sur une machine parallèle en sera donc plus aisé.

Le projet NOW[7] propose aussi une démarche visant à l'intégration dans ces systèmes UNIX des éléments logiciels permettant une coopération efficace entre les différents nœuds. Des travaux en cours depuis plusieurs années sur GLUnix (Global Layer Unix) concernent la migration automatique des processus, la répartition de la gestion des systèmes de fichiers, les techniques de « swap distribué » ainsi que tous les services dont peut bénéficier une machine parallèle dont les nœuds sont interconnectés de manière efficace.

1-1-6 La bibliothèque de programmation : spécifique et standard.

L'interface de programmation qu'un système parallèle propose à ses programmeurs doit répondre à deux critères opposés.

D'un côté, elle doit être la plus proche possible des mécanismes matériels d'échange de messages pour pouvoir bénéficier au niveau applicatif du maximum de performances. Ceci implique le développement d'une bibliothèque spécifique, qui doit tirer parti du réseau d'interconnexion. L'un des mécanismes qui semble être le plus efficace pour tirer parti des fortes performances d'un réseau d'interconnexion repose sur les « Active Messages »[8].

D'autre part, des interfaces de programmation standards ont été définies, pour permettre un portage instantané des applications existantes. PVM ou MPI[9], par exemple, doivent faire éventuellement l'objet d'une implémentation spécifique pour une architecture, de manière à

[7] [PATTER94]
[8] [EICKEN] et [MARTIN]
[9] [MPI] et [PACHE]

- Une bibliothèque de programmation : celle-ci propose une interface logicielle qui permet au programmeur d'applications d'utiliser les mécanismes de communication de la machine, et de dialoguer avec le ou les systèmes d'exploitation.

Dans les sections suivantes, nous présentons les évolutions actuelles des recherches sur les machines parallèles, qui ont mené à la standardisation d'au moins deux de ces quatre éléments.

1-1-4 La cellule monoprocesseur : une station de travail du marché.

Un problème souvent rencontré avec les machines parallèles est illustré par David Patterson[6] . En analysant la date de sortie sur le marché d'une station de travail monoprocesseur et la date de sortie d'une machine parallèle utilisant le même microprocesseur, on peut remarquer un décalage de plusieurs mois, voire de plusieurs années (figure 1-b).

Machine parallèle	Processeur	Année pour une station équivalente.	Disponibilité de la machine parallèle.
T3D	150 Mhz Alpha	1992-1993	1993-1994
Paragon	50 Mhz i860	1991	1992-1993
CM-5	32 Mhz Sparc 2	1989-1990	1991-1992

Figure 1-b : machine parallèle et station de travail incluant le même processeur.

Avec une augmentation des performances de 50% par an, ceci implique un retard systématique des cellules monoprocesseurs des machines parallèles sur les stations de travail.

Ce phénomène va certainement s'atténuer dans le futur. Certains constructeurs de machines parallèles, comme IBM par exemple, sont aussi les fabricants de leur propre microprocesseur. Ils sont alors capables de développer simultanément la station de travail et le nœud de la machine parallèle utilisant le nouveau processeur.

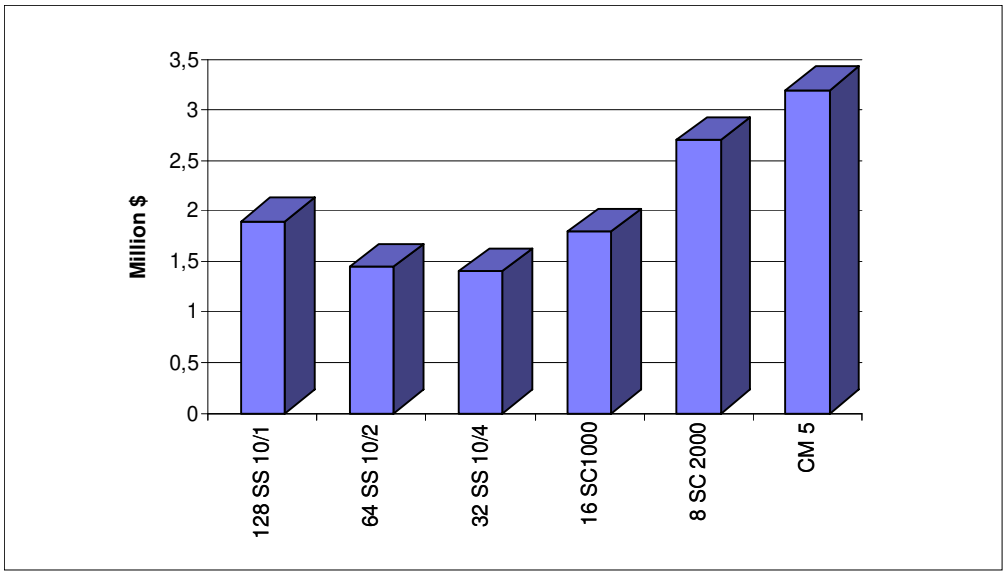

Figure 1-c : comparaison du prix de différentes solutions équivalant un CM-5.

Autre inconvénient, la différence de prix se fait également ressentir. Dans la figure 1-c, Patterson compare les prix respectifs de différents systèmes totalisant la même puissance qu'un CM-5 à 128 processeurs, en incluant le coût du réseau d'interconnexion. Cette différence est

[6] [PATTER94] et [PATTER95]

1-1-2 Pourquoi concevoir des machines parallèles ?

La puissance des architectures monoprocesseurs croît régulièrement. Cependant, un certain nombre d'applications nécessitent plus de puissance que les machines monoprocesseurs sont capables d'en fournir. On peut distinguer deux types de besoins.

Les machines génériques :

Certaines applications (météorologie, exploration pétrolière, nucléaire civile et militaire) nécessitent une importante puissance de calcul. Le coût des machines permettant de résoudre ces problèmes est particulièrement élevé (parfois plusieurs millions de francs).

La diversité des problèmes ou les recherches en cours sur les algorithmes permettant de résoudre ces problèmes nécessitent l'emploi de machines parallèles génériques, reprogrammables, qui puissent être utilisées dans différents contextes.

Les fabricants s'emploient donc à proposer des ordinateurs possédant la plus grande puissance de calcul, et la plus forte souplesse d'utilisation possible.

Les machines dédiées :

Un autre type de machine est souvent utilisé pour résoudre des problèmes spécifiques.

Thomson-Sintra ASM, par exemple, construit des systèmes radar. Différents capteurs acquièrent de très grandes quantités de données. Ces données sont ensuite traitées en deux étapes. Dans un premier temps, des opérations spécifiques de traitements de signal sont effectuées par des processeurs spécialisés, qui transmettent, dans un second temps, leurs résultats à des microprocesseurs classiques qui achèvent le traitement.

La nature des applications (militaire, aviation civile) impose que ces opérations soient effectuées en temps réel. La complexité de ces opérations a rendu indispensable la construction de machines parallèles.

Le CERN construit des dispositifs similaires pour traiter les résultats obtenus lors des collisions de particules dans les accélérateurs.

La particularité de ces deux applications rend indispensable la construction de machines spécialisées, les machines génériques n'offrant pas les performances nécessaires au juste coût.

1-1-3 Qu'est-ce qu'une machine parallèle ?

Une machine parallèle peut se décomposer en quatre entités :

- Une cellule monoprocesseur : celle-ci est répliquée un certain nombre de fois, elle contient tous les éléments nécessaires au fonctionnement du microprocesseur (RAM, caches, gestionnaire d'interruptions, etc.).
- Un mécanisme de communication : c'est-à-dire le système matériel qui relie ces différentes cellules ainsi que le ou les protocoles de communication. Cela peut être un bus ou un réseau d'interconnexion point-à-point.
- Un système d'exploitation qui gère l'ensemble du système : il se peut qu'un seul système d'exploitation, centralisé ou réparti, orchestre tous les processeurs, ou bien que chacun d'entre eux possède son propre système d'exploitation.

Chapitre 1 Problématique.

Le but de ce chapitre est de définir clairement nos objectifs, de positionner notre approche, et de mettre en lumière les problèmes que nous avons à résoudre et auxquels nous apporterons des solutions.

1-1 Présentation du contexte.

1-1-1 Présentation de la structure classique des ordinateurs.

Nous présentons sur la figure 1-a l'architecture des ordinateurs monoprocesseurs répandue aujourd'hui. Elle se compose, en général, d'au moins 4 éléments.

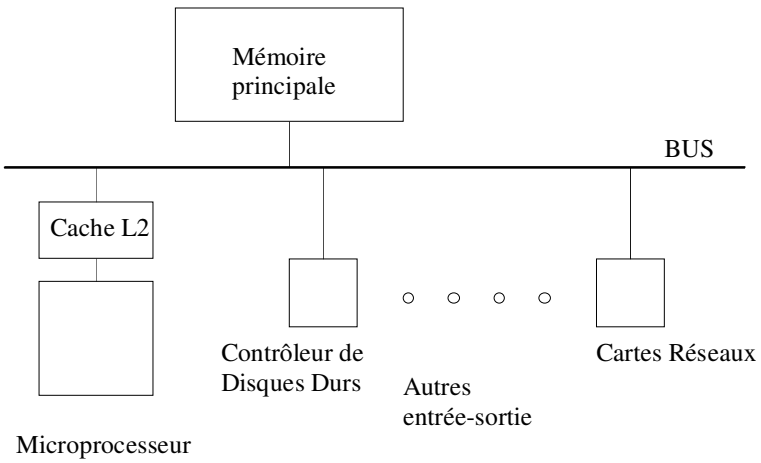

Figure 1-a : structure classique des ordinateurs.

- Le microprocesseur. Il exécute des programmes contenus dans la mémoire. Il dispose généralement d'un mécanisme de gestion de mémoire virtuelle (MMU) et d'un support matériel pour la gestion de processus qui lui permet d'exécuter, en temps partagé, différentes tâches. Il contient également un ou plusieurs caches de premier niveau.
- Le cache de second niveau. C'est une fraction de mémoire plus rapide et plus coûteuse que la mémoire principale, qui contient un sous-ensemble des données de celle-ci, que le processeur a dernièrement manipulées. En partant du principe que le processeur manipule plusieurs fois de suite les mêmes données, elle permet de raccourcir le temps moyen d'accès du processeur à une donnée de la mémoire principale.
- La mémoire principale. Celle-ci contient aussi bien les programmes que les données que ces derniers manipulent. Elle est d'un accès relativement lent par rapport aux performances du processeur.
- Un bus système. Il permet au processeur d'accéder à la mémoire ainsi qu'à des périphériques divers, disques durs, contrôleurs réseaux etc.

nous présentons en particulier le mécanisme fondamental de notre réseau, c'est-à-dire le « Direct Deposit Stateless Receiver Protocol ».

La machine MPC, proposant une mise en œuvre de ces deux composants spécifiques, ainsi que des mécanismes logiciels qui permettent d'accéder au réseau sont présentés dans le chapitre 5.

Dans le sixième chapitre, nous présentons une évaluation des performances attendues.

Les projets RCUBE, PCI-DDC et MPC.

Le projet **RCube** constitue la première partie de notre démarche. Il s'appuie fortement sur la technologie de liaison série point-à-point développée par l'équipe de Roland Marbot chez Bull[1] . Ce projet s'effectue dans le cadre de l'équipe Architecture du laboratoire MASI[2] .

En 1992, Belkacem Zerrouk lance l'idée d'un routeur faible latence et haut débit autour de ce lien et effectue les premières études d'architecture et de réalisation. En 1994, Vincent Reibaldi[3] , Belkacem Zerrouk et moi-même faisons aboutir le projet et réalisons le routeur RCube. Dès cette époque, ce composant est conçu pour répondre aux exigences des fabricants de machines parallèles. Ce projet est financé par la CEE dans le cadre du projet Esprit OMI-HIC.

Parallèlement, un outil de simulation et d'estimation de performance pour les réseaux d'interconnexion est mis au point par Abdelhafid Bouaraoua[4] et Belkacem Zerrouk, dans le cadre du projet Esprit OMI-Macrame.

A la fin de l'année 1994, l'idée de développer, au sein du MASI, un réseau d'interconnexion complet pour machines parallèles se basant sur RCube est acquise. La définition d'un second composant, **PCI-DDC**, pour la réalisation de l'interface réseau apparaît comme l'élément indispensable. Le prototypage commence alors avec Jean-Jacques Lecler et moi-même, puis continue en 1995, avec Jean-Lou Desbarbieux, Frank Wajsbürt et moi-même[5] . Le développement de ce composant est financé par la CEE dans le cadre du projet Esprit EUROPRO, en coopération avec Thomson.

Des partenaires comme Bull, Parsytec, INMOS, GEC, Thomson ont participé à ce projet, soit en apportant un soutien technique, soit en apportant leur expertise, soit en participant à la spécification.

Le projet **MPC** vise la réalisation d'une machine parallèle. Il s'agit de mettre en œuvre les deux composants développés par notre équipe pour montrer leurs performances et la validité de nos choix technologiques. Un deuxième objectif est de fournir une plate-forme d'expérimentation pour les développements logiciels sur les machines parallèles en proposant un système à bas coût, accessible aux équipes universitaires.

Plan du mémoire.

Dans le premier chapitre, nous définissons la problématique et précisons notre approche, en limitant clairement notre ambition, en mettant en avant les enjeux et les problèmes que nous nous proposons de résoudre.

Dans le chapitre deux, nous présentons l'architecture de plusieurs réseaux d'interconnexion de machines parallèles. Certaines sortent du cadre de notre approche, mais nous ont apporté des solutions originales à des problèmes qui se présentaient dans notre étude; d'autres, qui rentrent plus directement dans notre cadre, proposent aux problèmes rencontrés des solutions différentes.

Ensuite, nous présentons l'architecture des deux éléments de notre réseau : Le routeur RCube dans le chapitre 3, et le composant d'interface PCI-DDC dans le chapitre 4. Dans ce dernier,

---

1 [HSL].
2 Méthodologie et Architecture des Systèmes Informatiques, Université Paris VI..
3 [REIBALDI].
4 [BOUARA].
5 Voir en annexe 1 la liste des personnes impliquées dans le projet MPC et leur contribution.

Introduction.

« La recherche d'alternative à la structure classique débuta au milieu des années 60, quand la loi du rendement non proportionnel commença à prendre effet dans l'effort d'amélioration de vitesse de fonctionnement des ordinateurs.[...] La vitesse de fonctionnement des circuits électroniques est limitée en dernier ressort dans leur vitesse de fonctionnement par la vitesse de la lumière [...] et beaucoup de circuits fonctionnaient déjà dans la gamme des nanosecondes. »

Bouknight et Al. [1972]

La course à la performance des ordinateurs modernes s'est orientée ces dernières années dans deux directions complémentaires. D'un côté, l'amélioration des performances des ordinateurs à architecture classique, construits selon le modèle de Von Neuman, s'est poursuivie, conduisant à une augmentation de la puissance et de la complexité des monoprocesseurs. D'autre part, des alternatives à la structure classique se sont développées, principalement les machines parallèles, où plusieurs ordinateurs « Von Neuman » coopèrent à l'exécution d'une tâche.

Les enjeux de cette course ont une incidence directe sur notre mode de vie. L'augmentation de la puissance des ordinateurs influe sur l'économie, la vie sociale, l'environnement et la politique. L'une des conséquences de ce mouvement est que les moyens techniques et humains mis au service de cette course par les entreprises ont décuplé.

Cette course à la performance s'accompagne d'un mouvement de standardisation autour d'un petit nombre de composants, entraînant une baisse vertigineuse de prix des unités centrales. Ces ordinateurs, qui sont techniquement et financièrement irréalisables par une équipe universitaire, permettent de concevoir des architectures plus complexes, parallèles, en utilisant comme des « briques » les ordinateurs monoprocesseurs produits par l'industrie. Cette approche a tendance à se répandre, comme nous pouvons le voir dans le chapitre 2.

Cependant une machine parallèle n'est pas seulement une collection de machines à un seul processeur. Il faut que celles-ci puissent échanger des informations, se synchroniser pour coopérer, et il faut que ces opérations ne nécessitent qu'une fraction de leurs ressources, sans quoi l'approche « parallèle » n'a aucun sens.

L'un des éléments principaux permettant cette coopération entre les machines est le réseau d'interconnexion. Il permet l'acheminement de données entre les différents nœuds. Ces données sont à la fois celle manipulées par les programmes, les programmes eux-mêmes ainsi que les éléments de signalisation que ceux-ci s'échangent.

Cette thèse présente l'architecture d'un mécanisme de communication qui permet la coopération entre les différents processeurs. Notre objectif est de définir et de concevoir le « minimum » à rajouter à ces processeurs isolés pour les faire collaborer, tout en gardant de bonnes performances. En aucun cas nous ne tentons de modifier ces cellules monoprocesseurs ou leur accès aux périphériques.

Cette approche doit permettre de fournir l'un des outils indispensables à la construction des machines parallèles : un réseau d'interconnexion à haute performance.

FIGURE 4-O : FILE D'ATTENTE D'EMISSION ET ALIGNEMENT. 75
FIGURE 4-P : FILE D'ATTENTE DE RECEPTION ET ALIGNEMENT. 76
FIGURE 4-Q : CHARGEMENT DES TABLES DE RCUBE PAR PCI-DDC. 77
FIGURE 4-R : AUTOMATE PRINCIPAL DE PCI-DDC. 78
FIGURE 5-A : TAXINOMIE DES MACHINES PARALLELES. 81
FIGURE 5-B : PRESENTATION SYNTHETIQUE DE NOTRE ARCHITECTURE. 83
FIGURE 5-C : ARCHITECTURE GLOBALE DE L'INTERFACE LOGICIELLE. 84
FIGURE 5-D : SYNOPTIQUE D'UN TRANSFERT COMPLET. 86
FIGURE 5-E : VERROUILLAGE. 88
FIGURE 6-A : AUTOMATE DE PCI-DDC. 93
FIGURE 6-B : CHARGE DES LIENS EN FONCTION DE LA TAILLE DES PAGES. 94
FIGURE 6-C : LATENCE D'UN TRANSFERT EN FONCTION DE LA TAILLE DES PAGES. 95
FIGURE 6-D : GENERATION DE PAQUETS DANS LA SIMULATION MILE. 96
FIGURE 6-E : GRILLE COMPOSEE A DEUX DIMENSIONS. 97
FIGURE 6-F : LATENCE DU TRANSFERT D'UN PAQUET POUR DEUX TAILLES DE RESEAU. 97
FIGURE 6-G : DETAIL DE LA COURBE PRECEDENTE. 98
FIGURE 6-H : EVALUATION DU GAIN DE L'ADAPTABILITE SUR UNE GRILLE. 99
FIGURE 6-I : SIMULATION AVEC PCI-DDC EN RECEPTION PRIORITAIRE. 100
FIGURE 6-J : SIMULATION AVEC PCI-DDC EN PRIORITE TOURNANTE. 101
FIGURE 6-K : LATENCE DE ROUTAGE ET TAILLE DE PAQUET. 102
FIGURE 6-L : COMPARAISON RCUBE / ETAT DE L'ART (1). 103
FIGURE 6-M : COMPARAISON RCUBE / ETAT DE L'ART (2). 104
FIGURE 6-N : COMPARAISON PCI-DDC / ETAT DE L'ART (1). 104
FIGURE 6-O : COMPARAISON PCI-DDC / ETAT DE L'ART (2). 105

Illustrations

FIGURE 1-A : STRUCTURE CLASSIQUE DES ORDINATEURS. 11
FIGURE 1-B : MACHINE PARALLELE ET STATION DE TRAVAIL INCLUANT LE MEME PROCESSEUR. 13
FIGURE 1-C : COMPARAISON DU PRIX DE DIFFERENTES SOLUTIONS EQUIVALANT UN CM-5. 13
FIGURE 2-A : MATRICE TRIDIMENSIONNELLE. 20
FIGURE 2-B : TORE A TROIS DIMENSIONS. 20
FIGURE 2-C : STRUCTURE INTERNE DU ROUTEUR. 21
FIGURE 2-D : CANAUX VIRTUELS DE COMMUNICATION. 22
FIGURE 2-E : GESTION DE L'INTERBLOCAGE DANS LES RESEAUX A UNE SEULE MEMOIRE TAMPON. 22
FIGURE 2-F : STRUCTURE DES NŒUDS DU CRAY T3D. 23
FIGURE 2-G : GENERATION DES CARACTERES « STOP » ET « GO ». 27
FIGURE 2-H : FORMAT DES PAQUETS MYRINET. 28
FIGURE 2-I : LE ROUTEUR MYRINET. 28
FIGURE 2-J : L'INTERFACE LANAI. 29
FIGURE 2-K : RECOPIES MEMOIRE DANS LES TRANSFERTS TCP ET UDP. 30
FIGURE 2-L : MOT DU DS-LINK. 32
FIGURE 2-M : FORMAT DES PAQUETS. 33
FIGURE 2-N : ROUTAGE WORMHOLE. 33
FIGURE 2-O : ROUTAGE EXPLICITE. 34
FIGURE 2-P : ROUTAGE PAR INTERVALLES. 34
FIGURE 2-Q : ROUTAGE PAR GROUPES. 35
FIGURE 2-R : ARCHITECTURE DU SP2. 36
FIGURE 2-S : TRANSFERT DE PAQUETS. 37
FIGURE 2-T : ARCHITECTURE INTERNE DU ROUTEUR VULCAN. 38
FIGURE 2-U : ARCHITECTURE DU CONTROLEUR MCA DU SP2. 39
FIGURE 3-A : CODAGE 9 BITS VERS 12 BITS. 42
FIGURE 3-B : INVERSION. 43
FIGURE 3-C : FORMAT DU PAQUET RCUBE. 43
FIGURE 3-D : PROTOCOLES. 44
FIGURE 3-E : LE ROUTEUR RCUBE. 46
FIGURE 3-F : ROUTAGE WORMHOLE. 47
FIGURE 3-G : LE ROUTAGE PAR INTERVALLES. 47
FIGURE 3-H : SUPPRESSION DE L'EN-TETE ET SOUS-RESEAUX. 49
FIGURE 3-I : SCHEMA BLOC DE RCUBE. 50
FIGURE 3-J : SCHEMA BLOC DE R3P. 51
FIGURE 3-K : ROUTAGE D'UN PAQUET : INTERACTION ENTRE LES BLOCS. 52
FIGURE 3-L : SELECTION DE L'INTERVALLE. 53
FIGURE 3-M : SELECTION D'UN LIEN DE SORTIE. 53
FIGURE 3-N : COMPACTAGE DES PAQUETS. 54
FIGURE 3-O : MECANISME DE CONTROLE DE FLUX. 56
FIGURE 3-P : LE SELECTEUR DE PRIORITE TOURNANTE. 56
FIGURE 4-A : ARCHITECTURE D'UN MONOPROCESSEUR A BUS PCI. 59
FIGURE 4-B : RECOPIE DANS LES PROTOCOLES CLASSIQUES. 61
FIGURE 4-C : DIRECT DEPOSIT PROTOCOL. 62
FIGURE 4-D : GESTION DE LA LISTE DES PAGES EN EMISSION. 65
FIGURE 4-E : FORMAT DE LA LPE. 66
FIGURE 4-F : FORMAT DES PAQUETS. 67
FIGURE 4-G : FORMAT DE LA LMR. 68
FIGURE 4-H : EXEMPLE DE RECEPTION. 68
FIGURE 4-I : LECTURE DISTANTE. 70
FIGURE 4-J : SCHEMA BLOC DE PCI-DDC. 71
FIGURE 4-K : LES TROIS HORLOGES DE PCI-DDC. 71
FIGURE 4-L : COOPERATION DE LA FILE D'ATTENTE ASYNCHRONE ET DU BANC DE REGISTRES A L'EMISSION. 72
FIGURE 4-M : GESTION DU CONTROLE DE FLUX. 73
FIGURE 4-N : CAS GENERAL DE DESALIGNEMENT EMETTEUR / RECEPTEUR. 74

4-3-5 Protocole PCI. 76
4-3-6 L'automate de contrôle. 77
4-3-7 Gestion des erreurs. 78

4-4 Conclusion. 79

CHAPITRE 5 : UN EXEMPLE D'UTILISATION : MPC. 81

5-1 La machine MPC. 81
5-1-1 Positionnement. 81
5-1-2 Le réseau d'interconnexion. 82
5-1-3 Architecture. 83

5-2 L'interface logicielle Send / Receive. 84
5-2-1 Présentation. 84
5-2-2 Interface « PUT ». 85
5-2-3 Interface « SR-P ». 86
5-2-4 Interface « SR-V ». 88
5-2-5 Détournement de l'interface « Socket ». 89

5-3 Conclusion. 90

CHAPITRE 6 ANALYSE DES PERFORMANCES. 91

6-1 Conditions expérimentales. 91
6-1-1 Objectifs. 91
6-1-2 Mode opératoire. 91

6-2 Résultats. 92
6-2-1 Présentation de la simulation PCI-DDC. 92
6-2-2 Débit et latence de PCI-DDC. 93
6-2-3 Présentation de la simulation d'un réseau de RCube. 96
6-2-4 Débit et latence d'un réseau RCube. 96
6-2-5 Evaluation du gain dû à l'adaptabilité. 98

6-3 Optimisations. 99
6-3-1 Priorités entre l'émission et la réception. 99
6-3-2 Limitation de la taille des paquets. 101
6-3-3 Profondeur des files d'attente. 102

6-4 Comparaison. 103
6-4-1 Eléments de routage. 103
6-4-1 Interface et protocole. 104

6-5 Conclusion. 105

CONCLUSION. 107

REMERCIEMENTS. 109

BIBLIOGRAPHIE. 110

2-3-5 En résumé. 36

2-4 IBM SP2 **36**
2-4-1 Architecture. 36
2-4-2 Le lien de communication. 37
2-4-3 Le routeur Vulcan. 37
2-4-4 Le système d'interface SP2. 38
2-4-5 En résumé. 40

2-5 Conclusion. **41**

CHAPITRE 3 : LE COMPOSANT DE ROUTAGE RCUBE. **42**

3-1 Motivation. **42**
3-1-1 La liaison point-à-point HSL. 42
3-1-2 Paquets de taille quelconque. 43
3-1-3 Routage Wormhole. 44

3-2 Spécification. **46**
3-2-1 Présentation de RCube. 46
3-2-2 Contrôle de flux. 46
3-2-3 Routage par intervalles. 47
3-2-4 Une table de routage par entrée. 48
3-2-5 Header Deletion. 48
3-2-6 Adaptabilité. 49

3-3 Architecture interne. **50**
3-3-1 Architecture générale. 50
3-3-2 Synoptique d'un routage. 51
3-3-3 Configuration. 55
3-3-4 Contrôle de flux. 55
3-3-5 Adaptabilité et équité. 56
3-3-6 Gestion des erreurs. 57
3-3-7 Reset. 57

3-4 Conclusion **58**

CHAPITRE 4 : LE COMPOSANT D'INTERFACE PCI-DDC. **59**

4-1 Motivation **59**
4-1-1 Le bus PCI. 59
4-1-2 Les interfaces classiques. 61
4-1-3 « Direct Deposit Protocol ». 62
4-1-4 « Stateless Receiver Protocol ». 62
4-1-5 Conséquences. 63

4-2 Spécification. **64**
4-2-1 L'interface au bus PCI. 64
4-2-2 Méthode d'émission. 65
4-2-3 Méthode de réception. 68
4-2-4 La barrière de synchronisation. 69
4-2-5 Lectures distantes. 69

4-3 Architecture Interne. **70**
4-3-1 Présentation. 70
4-3-2 Interface vers le réseau HSL. 71
4-3-3 Gestion du flux. 73
4-3-4 Gestion de l'alignement. 74

Sommaire

SOMMAIRE 3

ILLUSTRATIONS 6

INTRODUCTION. 8

Les projets RCUBE, PCI-DDC et MPC. 9

Plan du mémoire. 9

CHAPITRE 1 PROBLEMATIQUE. 11

1-1 Présentation du contexte. 11
1-1-1 Présentation de la structure classique des ordinateurs. 11
1-1-2 Pourquoi concevoir des machines parallèles ? 12
1-1-3 Qu'est-ce qu'une machine parallèle ? 12
1-1-4 La cellule monoprocesseur : une station de travail du marché. 13
1-1-5 Le système d'exploitation : un Unix standard. 14
1-1-6 La bibliothèque de programmation : spécifique et standard. 14
1-1-7 Le médium : le lien de communication HSL. 15

1-2 Les objectifs. 16
1-2-1 Le débit. 16
1-2-2 La latence. 16
1-2-3 L'extensibilité. 16
1-2-4 Overhead. 17
1-2-5 Le coût. 17
1-2-6 La pérennité. 17

1-3 Conclusion. 18
1-3-1 Les performances. 18
1-3-2 Le coût. 18

CHAPITRE 2 ETAT DE L'ART. 19

2-1 Le CRAY T3D. 19
2-1-1 Le réseau d'interconnexion. 19
2-1-2 La cellule processeur. 22
2-1-3 Les ponts d'entrée-sortie et l'horloge centralisée. 24
2-1-4 En résumé. 25

2-2 Myrinet 25
2-2-1 Le lien parallèle. 26
2-2-2 Le routeur Myrinet. 27
2-2-3 L'interface LANAI. 29
2-2-4 En résumé. 30

2-3 INMOS T9000 C104 31
2-3-1 DS-Link. 32
2-3-2 Routage wormhole. 33
2-3-3 Routage explicite et par intervalles. 33
2-3-4 Adaptabilité par groupes. 35

A Daniel Mondot et Denis Roux

Thèse de Doctorat de l'Université Paris VI.

Spécialité Informatique.

Présentée par Frédéric Potter
Pour obtenir le grade de
Docteur de l'Université Paris VI.

Conception et réalisation d'un réseau d'interconnexion à faible latence et haut débit pour machines multiprocesseurs.

Soutenue le 11 Avril 1996,
devant le jury composé de

M. Daniel Etiemble	Rapporteur
M. Paul Feautrier	Rapporteur
Mme. Isabelle Demeure	Examinateur
M. Alain Greiner	Examinateur
M. Roland Marbot	Examinateur
M. Guy Mazare	Examinateur
M. Pascal Sainrat	Examinateur