Mettez un émulateur dans votre simulateur.

Comment accélérer l'évaluation de performances d'un réseau de files d'attente à l'aide d'une architecture reconfigurable.

Serge Martin, Vincent Olive - CNET /DTM
Cyril Labbé - M2000 *

France Telecom CNET, 28 rue du vieux Chêne, BP98 38243 Meylan cedex
serge.martin@cns.cnet.fr
M2000, Batiment Azur, 4 rue R.Razel, 91400 Saclay

Résumé. *L'émulation hardware est détournée de son objectif premier, la vérification de circuits, vers la simulation d'un réseau de files d'attente. L'objectif est d'évaluer la méthode sur une simulation à événements discrets, ou le parallélisme est dominant. Les résultats sont à la hauteur des prévisions : la rapidité des simulations autorise des campagnes de mesure beaucoup plus importantes que celles abordables par les simulateurs logiciels. Les machines à architecture reconfigurable évoluent rapidement : elles sont programmables par des langages de haut niveau, et ont des interfaces aussi sophistiqués que les débogueurs de simulateurs logiciels. Cette technologie peut sortir de l'usage exclusif de la conception des ASIC et trouver de nouvelles applications.*

Abstract. *Hardware emulation has changed from its primary purpose, i.e. circuit verification, towards simulation of queuing network. Our aim is to perform an evaluation of this simulation method on a discrete event process, in which parallelism is dominant. Results are up to expectations : the emulation speed allows for much longer simulations than those achieved by software simulators. Programmable architecture machines are changing rapidly : they are programmable using high level languages, and have interfaces which are as sophisticated as software debuggers. This technology can extend beyond the exclusive use of ASIC design to find new applications.*

Mots-clés : *architecture reconfigurable, simulation à événements discrets, réseau de files d'attente, machines parallèles, émulateur.*

Keywords: *reconfigurable hardware, discrete event simulation, queueing network, parallel computers, emulator.*

1. Introduction

Le développement de simulateurs performants peut s'effectuer par l'évaluation de nouvelles approches, rendues possibles en détournant des outils de leur utilisation courante. Grâce à l'amélioration significative des émulateurs hardware, nous proposons de montrer dans cet article que ces machines peuvent se transformer en puissants simulateurs. L'émulateur de Metasystems, de technologie très récente, est un bon exemple d'outil dont on peut élargir le champ d'application au-delà du domaine de la conception de circuits. L'application que nous décrivons est l'évaluation d'un commutateur à plusieurs étages, qui transmet des flux ATM, et dont il s'agit de déterminer les capacités en termes de taux de pertes de cellules. L'article montre que les résultats sont extrêmement plus fins que ceux des simulateurs logiciels actuels. Mais au-delà des résultats, nous avons cherché, dans cet article, à valoriser la méthode de simulation en insistant sur deux aspects. Premièrement, le domaine d'application : c'est le domaine des processus à événements discrets, également celui des méthodes particulaires. Nous avons détaillé la réalisation de notre modèle de réseau pour montrer comment formuler le problème avec ce type d'approche. Un autre bon exemple est donné par [3] sur l'exploration des bases de données de séquences génétiques. Les systèmes à base de réseaux de Petri, d'interconnexion de réseaux de neurones, devraient être aussi de bons candidats pour cette méthode.

Deuxièmement, nous avons voulu insister sur la mise en œuvre de la méthode. Même si ce n'est pas trivial, de réelles simplifications apparaissent avec l'utilisation de langages de haut niveau de description de hardware. Ces langages sont parfaitement adaptés à la description de processus concurrents, et mènent à un parallélisme massif de l'algorithme. La chaîne de compilation (synthèse de matériel) et l'outil d'émulation ont un inconvénient majeur: le coût. Il faut espérer que la progression de la technologie permettra de rendre accessible à un plus grand nombre ces architectures reconfigurables.

2. Présentation du matériel

2.1. *L'architecture*

Nous présentons une machine à architecture reconfigurable, en l'occurrence l'émulateur M500 de Metasystems [6]. Nous nous intéressons au travers de cette description aux caractéristiques typiques de ce genre d'architecture. Dans l'utilisation que nous en faisons, nous verrons que notre modèle utilise des spécificités propres à la machine M500. Une spécificité largement utilisée est la possibilité de déboguer: les nœuds internes peuvent être tracés et visualisés, et l'interface homme-machine offre toutes les bonnes caractéristiques d'un simulateur. La description qui suit sert à se faire une idée des possibilités de la

*Cyril Labbé prépare une thèse CIFRE, encadré par Frédéric Reblewski de la société M2000 et Jean-Marc Vincent maître de conférence à l'IMAG-LMC. Serge Martin et Vincent Olive sont ingénieurs chercheurs au CNET, centre de recherche de France Telecom.

machine M500 afin de cerner ses domaines d'applications. La description n'est pas exhaustive et on pourra se référer à d'autres descriptions [5, 4] pour mieux connaître la machine.

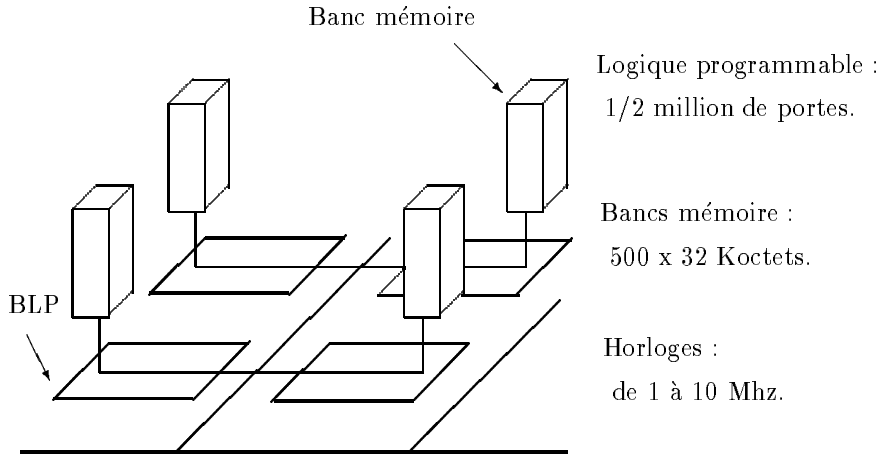

Figure 1: Vue schématique du M500. L'émulateur est composé de BLP, de blocs mémoire et d'un réseau d'interconnexions.

L'unité de base de l'émulateur M500 est le BLP (bloc de logique programmable); il a 4 entrées et une sortie, et peut être programmé pour réaliser toutes les fonctions booléennes: AND, OR, INVERTER, etc... Les BLP sont rassemblés dans des composants appelés META. Ce circuit, qui a été conçu spécialement par Metasystems, contient 128 BLP, ainsi que la logique de commande pour les programmer, et le réseau pour les interconnecter. En plus des fonctions booléennes, ou portes logiques, chaque BLP contient un élément mémoire qui peut servir de registre ou de latch. L'ensemble de ces éléments constitue la logique programmable. L'émulateur M500 contient un demi million de portes et 70 000 registres. A titre de comparaison dans le domaine des ASIC, cette configuration représente un circuit de 3 millions de transistors.

L'utilisateur de l'émulateur voit aussi des mémoires de types variés : synchrones, asynchrones, simple port, double port, sans restriction en adressage. La contenance totale est 17 Mo, mais la configuration des mémoires se fait par blocs: il y a 500 blocs de 32 000 octets que l'utilisateur groupe selon ses besoins. Les mémoires apportent un plus indéniable quand il s'agit de faire de la simulation. Elles permettent de stocker des résultats. On le verra, cette fonctionnalité est extrêmement importante car l'émulateur est cadencé à quelques MHz et il faut trouver un moyen de lire les nombreux résultats au vol. A cette cadence un lien avec une station de travail qui servirait de zone de stockage ne

suivrait pas.

La technologie de cette machine est déjà rendue obsolète par une nouvelle gamme qui multiplie ses capacités par 20. En fait l'évolution de ce genre de matériel suit celle des circuits numériques. C'est en matière de logiciels de synthèse et de routage qu'il faut s'inquiéter, car le temps d'exécution augmente linéairement avec la complexité des circuits.

Le réseau d'interconnexions doit être suffisamment dense pour utiliser au mieux la logique programmable disponible ainsi que les bancs mémoires. Compte tenu du nombre de portes logiques adressables, ce réseau est complexe et présente un inconvénient: la longueur des connexions sera toujours un frein à la rapidité. La structure du M500 est la suivante : les META sont assemblés sur des cartes, 24 META par carte, et 23 cartes sont disposées dans le rack du M500. Tant que le modèle émulé reste simple et utilise moins d'une carte, des vitesses de plusieurs MHz (jusqu'à 10 MHz) sont couramment atteintes. Dès que plusieurs cartes sont nécessaires la vitesse tombe entre 1 et 2 MHz.

Une caractéristique du M500 est qu'il est construit pour travailler de façon synchrone. D'une horloge maîtresse sont dérivées toutes les autres. Le système émulé doit de préférence respecter cette caractéristique, ce qui est souvent le cas pour les circuits que nous traitons au CNET. L'asynchronisme conduit à des difficultés car l'utilisateur ne maîtrise pas le routage de la logique programmable, donc les temps de propagation. Or c'est dans les interconnexions que les délais sont les plus importants. Il est donc préférable de s'en tenir à des modèles synchrones... ou bien attendre les prochaines versions de l'émulateur qui supportera des horloges indépendantes.

Enfin le M500 est pourvu d'entrées-sorties TTL, programmables en entrées, en sorties ou bidirectionnelles. Cet accès extérieur sera mis à profit dans notre expérimentation pour aller lire et écrire des données en cours d'exécution.

2.2. *La chaîne logicielle*

La production du code adapté à l'émulateur est identique à la conception des ASIC. Au CNET, nous privilégions le langage VHDL pour décrire un composant, nous utilisons la synthèse logique, puis la chaîne de compilation de l'émulateur sur les circuits programmables.

Pour les lecteurs qui ne sont pas familiers du domaine de la conception de circuits intégrés, nous donnons dans ce paragraphe un aperçu des nombreuses phases de la conception, avec les objectifs suivants: montrer qu'un émulateur est une machine massivement parallèle, montrer que l'on peut donner une description fine d'un problème pour l'exprimer en termes de logique booléenne, montrer que tout cela se fait avec des langages de haut niveau.

A l'origine d'une nouvelle génération d'outils de conception de circuits intégrés, il y a le langage VHDL [8, 7] (VHSIC Hardware Description Language), et d'autres langages similaires comme Verilog. Les langages ont été créés pour décrire un comportement simulable de circuits numériques. Ils supportent 2 niveaux de description : l'un fonctionnel, où les algorithmes se déroulent en parallèle; l'autre structurel, plus proche de l'architecture du

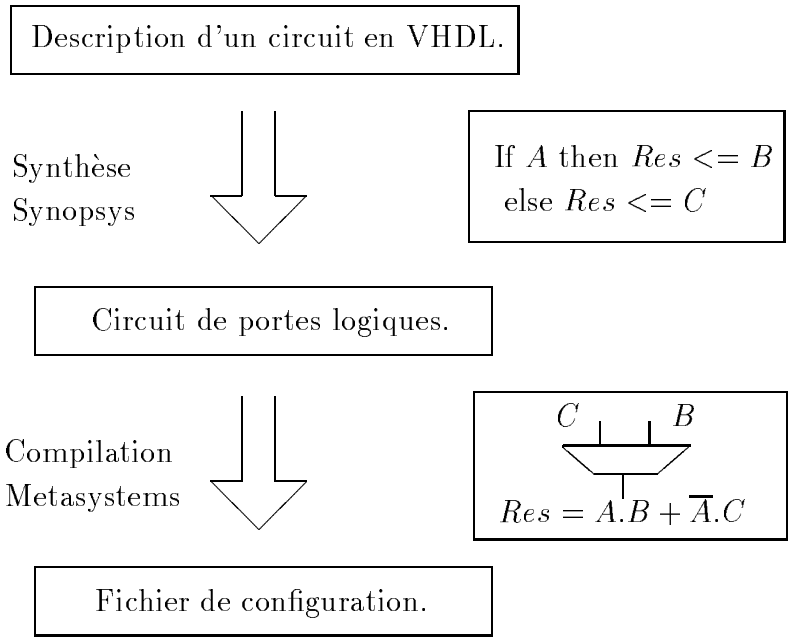

Figure 2: figure Chaîne logicielle.

système numérique (avec horloges, registres, mémoires). En attendant les progrès de la synthèse comportementale, nous avons choisi de programmer l'émulateur au niveau description structurelle. Ceci constitue un travail de recherche pour optimiser l'architecture de la machine spécialisée dans le traitement de la simulation des files d'attente.

Dans un circuit intégré numérique, il y a un ensemble de blocs fonctionnels, composés eux-mêmes de fonctions plus élémentaires. Au niveau le plus bas on trouve les constituants de la logique booléenne: les fonctions AND, OR, NAND, NOR, XOR, INVERTER, ou portes logiques. Les langages de conception permettent de structurer les fonctions à réaliser et d'améliorer la lisibilité. La figure 2 montre un exemple de multiplexeur et son expression en VHDL, en algèbre booléenne, et en un schéma logique. La lisibilité est nettement en faveur du langage VHDL. Plus on augmente la complexité de la fonction booléenne, plus le langage apporte de clarté et de lisibilité.

La séquentialité des traitements est obtenue par des éléments de mémorisation ou registres. A chaque cycle d'horloge, un calcul booléen est effectué, puis le résultat est mémorisé dans les registres. La valeur du registre est à son tour utilisé pour le calcul pendant le cycle suivant. Le point de synchronisation, donné le plus souvent par une horloge, est exprimé en VHDL par une instruction

spécifique, l'instruction wait.

A ce stade il est important de remarquer que la programmation est concurrente, c'est à dire qu'elle décrit des processus multiples qui se déroulent en parallèle et sont synchronisés par une horloge ou tout autre signal. Par comparaison aux langages informatiques parallélisables, où ce sont les compilateurs qui produisent le code adapté à l'exécution sur des processeurs parallèles, le langage VHDL décrit l'architecture de l'application au niveau le plus fin. Il faut souligner ici cet important aspect. On peut donc produire une description avec des processus massivement parallèles. Les composants hardware sont concurrents, et le langage qui les a pour cible doit posséder cette capacité. Avec VHDL (ou Verilog), on décrit des automates qui s'exécutent simultanément et ont des points de synchronisation.

La première étape de la conception consiste donc à utiliser un langage de description hardware pour décrire le composant ou le modèle. La conversion du code VHDL en portes logiques s'appelle la synthèse logique; c'est la première étape du flot de conception. Synergy (Cadence), Autologic (Mentor Graphics), Design Compiler (Synopsys) sont des outils de synthèse logique couramment utilisés.

La seconde étape consiste à placer le réseau de portes (netlist) sur le réseau de META de l'émulateur. Elle est nommée compilation dans l'environnement Metasystems, et produit un fichier de configuration. La compilation est dédiée à l'architecture du M500, elle utilise des logiciels développés spécialement par Metasystems pour optimiser les connexions entre les BLP, et attribuer la logique de commande nécessaire à la trace des signaux et au débogueur. Au moment de l'exécution sur l'émulateur, le fichier de configuration initialise des tables de vérité internes à chaque BLP pour programmer la fonction logique désirée. Le fichier de configuration initialise aussi le réseau d'interconnexion formé de matrices de commutation du type Crossbar. Il y a trois niveaux d'interconnexions: l'un est interne aux META et relie les 128 BLP entre eux, le deuxième assure l'interconnexion de tous les META d'un rack entre eux, le troisième permet l'assemblage de plusieurs racks (jusqu'à cinq M500 interconnectés).

2.3. *Contrôle de l'émulateur*

On décrit ici le moyen de lancer l'exécution du modèle configuré dans l'émulateur. Les moyens de contrôle sont en fait identiques au déroulement d'un programme sous le contrôle d'un débogueur. Tandis qu'un programme se déroule instruction par instruction, l'émulateur est contrôlé par son horloge. Le moniteur de l'émulateur permet de faire avancer l'horloge d'un nombre donné de périodes, ou alternativement d'une période à la fois. A chaque point d'arrêt, on peut examiner la valeur des signaux et des registres. Pour cette opération, le M500 garde la trace des 7000 dernières valeurs de tous les signaux. L'accès aux signaux se fait simplement par leur nom logique, extrait de la conception en VHDL. Le moniteur de l'émulateur effectue la correspondance entre le nom logique et l'emplacement physique du signal.

Par conséquent, les composants META sont plus complexes que les FPGA

(Field Programmable Gate Arrays). En plus de la logique de commande pour programmer les BLP, ils possèdent une logique de contrôle qui donne de réelles capacités de simulateur à l'émulateur. Le contrôle nécessite des ajouts sur la partie signaux d'interconnexions, nous citerons plus particulièrement:

- des chaînes de scan (registres pour le test) pour rapatrier les valeurs des signaux vers le moniteur,
- les mémoires associées à chaque signal et leur logique de stockage,
- les blocs logiques pour surveiller un signal et s'arrêter sur une valeur donnée,
- la possibilité d'aller changer la valeur d'un registre ou d'une entrée.

Les possibilités de contrôle sont utiles pour exploiter les résultats de la simulation. Le langage de commande de l'émulateur permet de créer des fenêtres pour visualiser les données de la simulation. Néanmoins nous avons également développé un lien VME (bus normalisé de 32 bits en parallèle) vers une station de travail. Un programme peut, par ce lien, échanger des données avec l'émulateur. En cours de simulation, les données s'affichent alors en temps réel.

3. Présentation du problème et choix des modèles

La simulation que nous présentons est celle d'un réseau de commutateurs ATM (Asynchronous Transfer Mode); l'objectif est de dimensionner les files d'attente afin de minimiser les pertes de cellules qui peuvent se produire quand toutes les cellules convergent vers un même récepteur. Sa file d'attente est alors saturée d'où il existe une certaine probabilité de pertes de cellules. Des simulateurs logiciels existent [1, 2], qui répondent au même objectif. Nous allons montrer qu'un simulateur matériel atteint des performances bien supérieures, et permet une observation très fine du trafic.

3.1. *Présentation du modèle*

Le commutateur a la structure présenté sur la figure 3.Nous avons réalisé un commutateur 4x4 à partir de deux étages de commutateurs 2x2. Chaque commutateur 2x2 possède une file d'attente sur chaque sortie pour enregistrer les cellules quand 2 cellules convergent au même moment sur la même sortie. Nous supposons que, à chaque instant, une ou plusieurs cellules sont consommées, c'est à dire transportées vers l'étage suivant. La consommation est un paramètre de la simulation.

3.2. *Modélisation du trafic*

Une liaison ATM se caractérise par le transport d'informations numériques, groupées dans des unités de base appelées cellules. Chaque cellule a 53 octets: 5 octets d'en-tête contiennent l'adresse de destination et diverses informations

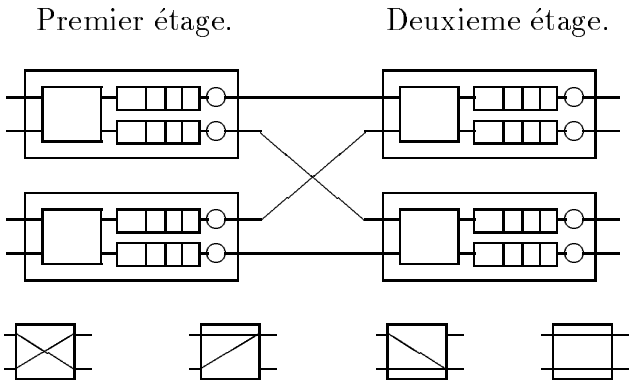

Figure 3: Structure du commutateur 4x4 composé de deux étages de commutateurs 2x2.

de contrôle; les 48 octets restant (payload) contiennent les données. La forme du trafic dépend de la source: le transport de la voix donne un trafic régulier, le transport de l'image ou de données provoque l'émission de cellules en rafales. Le processus des arrivées est modélisé par un processus géométrique, décrit ci-dessous, dont le paramètre ρ fixe la charge, c'est à dire la proportion des cellules utiles sur le débit maximum.

La modélisation d'un processus géométrique est issu d'un générateur aléatoire uniforme. A chaque slot ou coup d'horloge, la valeur aléatoire est comparée à la valeur de la charge ρ (par exemple 80%). Si elle est inférieure, une cellule est émise. On peut déterminer avec ces hypothèses la probabilité que plusieurs cellules soient émises consécutivement. Si i est la taille d'une rafale, sa probabilité est

$$P(i) = \rho(1-\rho)^{i-1}$$

La taille moyenne des rafales est donc:

$$A = \sum_{i=1}^{\infty} i * P(i) = 1/\rho$$

Dans notre simulation, les cellules ont un contenu. On y inscrit la destination, choisie de façon équiprobable.

Avec ces hypothèses on calcule exactement le taux de pertes au premier étage [9]. Par contre, il n'y a pas de solution par le calcul du taux de perte au deuxième étage. On approche simplement le résultat par encadrements stochastiques [9].

La modélisation du trafic est conditionné par l'obtention d'une loi aléatoire. Nous l'avons réalisée à l'aide d'un polynôme dont la période est très grande

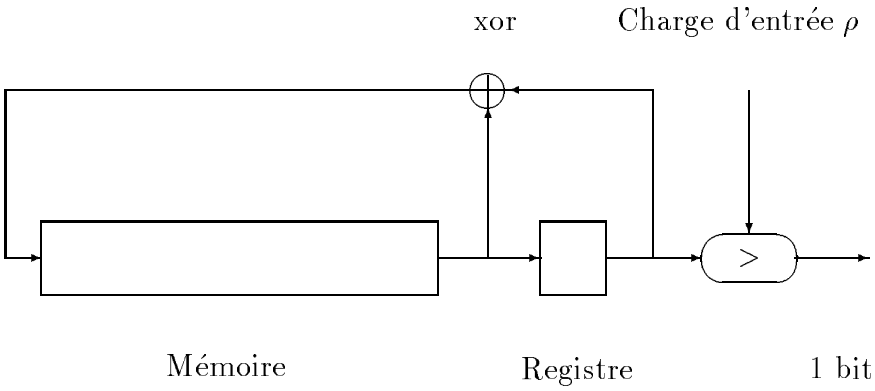

Figure 4: Générateur de motifs aléatoires.

devant le temps de simulation (sachant que l'on traite au moins 10^6 tirages par seconde) , ce qui se traduit par l'implémentation matérielle de la figure 4. La mémoire contient 126 valeurs de 16 bits. Un "ou exclusif" est fait entre la 126ème valeur et le registre, et le résultat est remis en mémoire après décalage. La valeur est comparée à la charge ρ pour déterminer l'émission d'une cellule. Ce schéma a l'avantage d'être économique en nombres de portes logiques, et est rendu facilement synthétisable grâce aux nombreuses mémoires disponibles sur le M500.

3.3. *Modélisation de l'architecture du commutateur*

Le commutateur reçoit des cellules avec un contenu. Celui-ci détermine de façon unique l'adresse de destination. La cellule est ensuite stockée dans une FIFO, avant d'être émise vers le commutateur suivant.

Il n'est pas indispensable qu'une cellule ait un contenu: à l'origine de nos simulations, le commutateur déterminait aléatoirement la destination. En fait, l'inscription d'une adresse dans une cellule permet plus de variantes dans les essais: il est plus aisé de créer des trafics complexes tels que des superpositions de trafic réguliers sur des trafics aléatoires, ou tels que des trafics avec destination préférentielle (hot spot).

A ce niveau de description on perçoit déjà les performances de la simulation matérielle. Un commutateur à 4 entrées chargées à 80%, fonctionnant à 1 MHz transmet un débit de 3.2 millions de cellules par seconde. En une heure, le réseau voit passer 10^{10} cellules, chiffre qui permet déjà d'évaluer des taux de pertes de 10^{-8}. Ces performances, qui sont comparables à celles d'un débit réel de 655 Mbit/s, sont très supérieures à un simulateur logiciel. On gagne en fait 5 ordres de grandeur , les simulateurs logiciels permettant habituellement

d'évaluer des taux de 10^{-3} avec une précision suffisante. La précision des résultats dépend du nombre de pertes. On calcule qu'à partir de 1000 cellules perdues, la précision des résultats est de 6%. Pour évaluer un taux de pertes de 10^{-8}, il faut donc faire passer environ 10^{11} cellules.

3.4. *Instrumentation*

L'analyse des résultats nécessite l'acquisition d'un certain nombre d'indices. En premier lieu les compteurs de cellules arrivées, les compteurs de cellules perdues. Sur l'émulateur, cela se traduit par de simples compteurs ou incrémenteurs. Notons cependant que pour compter le grand nombre de cellules, des mots de 32 bits ne suffiront pas. On utilisera des compteurs de 50 bits, ce que l'émulation hardware permet de réaliser aisément.

Nous avons développé également un analyseur de trafic, et mis en place un test du χ^2. On peut ainsi évaluer la perturbation du trafic après passage dans le commutateur, et le comparer au processus géométrique d'émission. Cette analyse consiste à compter les groupes de 1, 2, 3... cellules conjointes, ainsi que les groupes de cellules vides conjointes. Ces chiffres sont comparés aux valeurs théoriques. Ici encore, un certain nombre de compteurs sont nécessaires. Ces analyseurs sont ensuite disposés à la source, pour vérifier le bon fonctionnement du générateur aléatoire, puis au premier et au second étage.

Pour une approche plus dynamique de la simulation, nous avons développé une application indépendante des logiciels de contrôle de l'émulateur. Cette application lit en temps réel les données de la simulation à travers une carte VME reliée au SBUS de la station de travail, et met à jour le schéma du réseau reproduit dans une fenêtre. Au total 32 compteurs sont prélevés à la demande ou périodiquement, et les opérations de mise à jour des compteurs, graphes, taux et courbes dans la fenêtre, sont faites localement par l'application. Pour que les données lues soient cohérentes entre elles, l'horloge de la simulation est arrêtée pendant 32 cycles, le temps pour un automate de lire les données dans les mémoires et registres, et de les transférer sur le bus VME vers l'application.

3.5. *Évaluation de la complexité en machine*

Sur le M500, l'ensemble réseau et instrumentation occupe à peine deux cartes, sur les vingt deux disponibles. Nous avons donc implanté quatre fois le même circuit pour mener quatre expériences simultanées, en faisant varier les germes des générateurs aléatoires. Les résultats ont permis de réduire l'intervalle de confiance des résultats, sans diminuer la vitesse de fonctionnement.

Le temps de développement de cette application a été de quelques mois. Le réseau et l'instrumentation embarquée représentent quelques 3000 lignes de VHDL, et le pilotage temps réel à travers le bus VME nécessite à peu près autant de code C++. La difficulté vient du code VHDL qui est rarement réutilisable: une nouvelle version amène, presque à coup sûr, un changement d'architecture, et par conséquent une refonte complète du code.

4. Exemple de résultats et perspectives

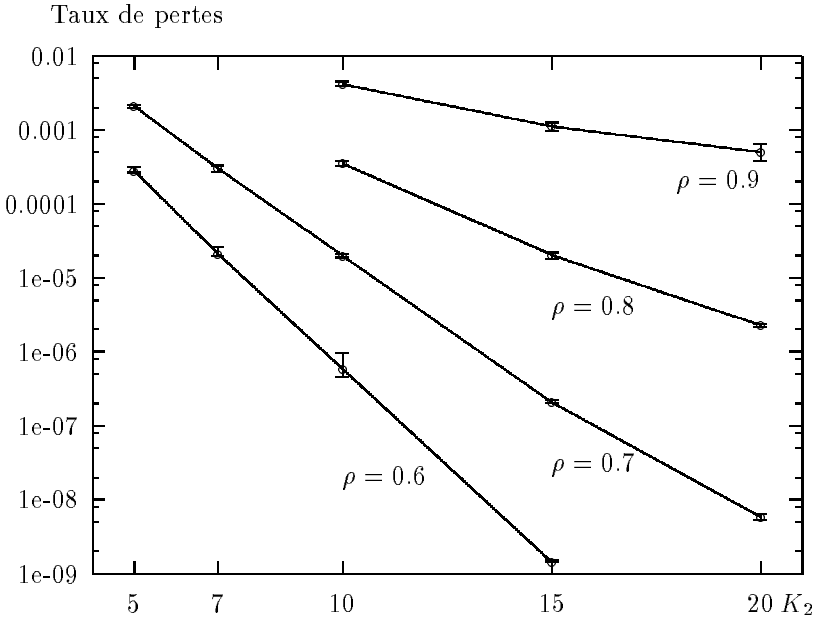

Figure 5: Taux de pertes sur l'ensemble du commutateur en fonction de la taille du deuxième étage (K_2). La taille du premier étage est fixée à 15. Trafic uniforme. Charge ρ variant de $0,9$ à $0,6$.

Le graphique de la figure 5 présente les taux de perte obtenus quand la charge du trafic varie. Le tampon du premier étage est fixe, alors que le tampon du deuxième étage varie de 5 à 20. Chaque point est une moyenne des résultats obtenus lors de plusieurs simulations. L'intervalle de confiance est représenté. Les portions droites, observées en début de chacune des courbes, s'expliquent par le fait que, lorsque la taille des tampons du deuxième étage est assez petite, toutes les pertes observées ont lieu au deuxième étage. Quand la valeur des tampons du deuxième étage dépasse celle du premier étage, alors les pertes se répartissent sur les deux étages, et la pente de la courbe change. Il faut remarquer les valeurs des pertes observées, allant de 10^{-2} à 10^{-9}, et l'intervalle de confiance élaboré à partir de plusieurs simulations. Ces résultats confirment le très bon comportement de ce simulateur.

5. Conclusion

Dans cet article nous avons montré qu'une machine à architecture reconfigurable pouvait se révéler un outil très puissant pour la simulation de processus

à événements discrets. L'application aux réseaux de files d'attente en est une illustration typique, où des gains en performance de 5 ordres de grandeur sont obtenus par rapport aux simulateurs logiciels. Mais au delà des résultats, c'est également la méthode que nous cherchons à divulguer. Dans ce but, l'article insiste sur deux aspects: premièrement, nous montrons les types d'application qui pourraient bénéficier d'un traitement par architecture reconfigurable, en s'appuyant sur un exemple décrit finement. Deuxièmement, nous montrons que la technologie des émulateurs et de la CAO font des progrès constants : la programmation se fait avec des langages de haut niveau, l'émulation devient aussi facile à piloter qu'un débogueur de programmes, bref la technologie n'est plus seulement l'affaire des concepteurs de matériel.

L'évaluation de nouvelles méthodologies pour la simulation résulte de la collaboration de trois équipes, contractualisée par une thèse CIFRE. Trois domaines sont ainsi rapprochés : l'émulation matérielle (F. Reblewski, M2000), l'évaluation de performances (J-M. Vincent, IMAG-LMC), l'application aux réseaux (CNET). La première partie de cette thèse nous a permis de faire la démonstration que la méthode donnait des résultats fiables, et que les performances étaient en accord avec les prévisions. Les travaux futurs porteront sur la recherche d'une meilleure caractérisation des trafics brassés afin de mieux les contrôler.

References

[1] *SES Workbench Reference Manual*, feb. 1992.

[2] M.Veran et D.Potier. QNAP2 : a portable environment for queueing network modeling. *Modeling Techniques and tools*, 1984.

[3] E.Lemoine et J.Sallantin. Un système reconfigurable dédié à la comparaison de séquence génétiques. *Rairo : technique et science informatiques*, 14-1, 1995.

[4] L.Burgun F.Reblewski G.Fenelon J.Barbier and O.Lepape. Serial fault emulation. In *Proceedings of the 33rd Design Automation Conference 1996 (DAC 96)*, pages 801–806, Metasystems, France, 1996.

[5] L.Burgun and F.Reblewski. Première génération d'émulateurs matériels meta-systems. In *Quatrième symposium sur les architectures nouvelles de machines*, Metasystems, France, 1996.

[6] Metasystems. *http://www.mentorg.com/products/simexpress/index.html.*

[7] R. Airiau, J.-M. Berge, and V. Olive. *VHDL du language à la modélisation.* Presses polytechniques et universitaires romandes, France Telecom, 1990.

[8] R. Airiau, J.-M. Berge, and V. Olive. *Circuit Synthesis with VHDL.* Kluwer Academic Publishers, France Telecom, 1994.

[9] L. Truffet. *Méthodes de Calcul de Bornes Stochastiques sur des Modèles de Systèmes et de Réseaux.* PhD thesis, Université Paris VI, 1995.