Algorithmes, machines et langages

M. Gérard BERRY, membre de l'Institut
(Académie des sciences), professeur

ENSEIGNEMENT : LE TEMPS ET LES ÉVÉNEMENTS EN INFORMATIQUE

J'ai donné le cours « Le temps et les événements en informatique » dans le cadre de la chaire Algorithmes, machines et langages, créée le 4 juillet 2013 comme première chaire de plein exercice en informatique. J'avais introduit l'informatique au Collège de France en 2007-2008 par le cours « Pourquoi et comment le monde devient numérique [1] », au sein de la chaire annuelle d'Innovation technologique Liliane Bettencourt, puis, en 2009-2010, par le cours « Penser, modéliser et maîtriser le calcul informatique [2] » en inauguration de la chaire annuelle Informatique et sciences numériques, créée en partenariat avec Inria.

Mon premier cours proposait une présentation générale de la discipline qu'est l'informatique et de ses façons de penser. Il était en forme de leçons de choses sur différents sujets centraux dans le domaine : algorithmique, circuits, programmation, réseaux, traitement d'images, etc. Mon deuxième cours, plus technique, étudiait les fondements mathématiques et algorithmiques de la notion de calcul, en insistant sur ses formalisations et sur l'importance de *programmer juste et le démontrer*. Mes cours sur la chaire Algorithmes, machines et langages continueront dans cette veine, en s'intéressant plus spécifiquement aux problèmes liés au temps et aux événements en informatique. Bien qu'essentiel dans l'informatique ubiquitaire moderne, ce sujet reste étonnamment peu traité dans la communauté : l'algorithmique en parle peu, et les langages de programmation classiques n'en font mention que du bout des lèvres.

Le cours résumé ici a mis en place le sujet et présenté des formalisations et résultats spécifiques à des sous-problèmes centraux. J'ai surtout présenté en profondeur les méthodes et langages synchrones qui occupent mon travail depuis 30 ans. Le cours de 2013-2014 sera consacré aux systèmes beaucoup plus répartis géographiquement et logiquement, qui se généralisent en informatique. Je prévois de consacrer les années suivantes à la preuve de correction des programmes

1. http://www.college-de-france.fr/site/gerard-berry/course-2007-2008.htm.
2. http://www.college-de-france.fr/site/gerard-berry/course-2009-2010.htm.

temporels et événementiels, sujet essentiel car ils sont au cœur de systèmes dont la sûreté de fonctionnement est critique.

Le cours de cette année s'est composé de la leçon inaugurale et de huit séances. La première séance, de deux heures, a été consacrée à une présentation générale des problèmes posés, puis au fonctionnement temporel des circuits digitaux combinatoires. Les six séances suivantes, chacune composée d'un cours d'une heure suivi d'un séminaire donné par une personnalité extérieure, ont examiné différents aspects du domaine. La dernière séance a été constituée de deux séminaires sur un sujet où nos méthodes ont eu des apports inattendus, la composition et l'interprétation musicale. Les séminaires sont pour moi aussi importants que les cours, car ils y apportent des compléments et des points de vue différents. J'en résumerai donc le contenu. Cours et séminaires ont été filmés : les vidéos sont visibles sur le site du Collège de France [3], également en traduction anglaise [4].

Leçon inaugurale

Dans ma leçon inaugurale du 28 mars 2013 [5], j'ai présenté la problématique de mon cours de 2013-2014 et de ceux qui suivront. Les algorithmes, machines et langages dont je souhaite parler sont liés à la gestion du temps et des événements en informatique, avec cinq grands domaines d'applications :

– les *logiciels embarqués*, en particulier ceux qui assurent le contrôle en temps réel de systèmes physiques en automobile, avionique, etc. :

– les *circuits électroniques*, autrefois relativement simples, mais devenus tellement gigantesques en nombre d'éléments architecturaux et de transistors qu'on les appelle « systèmes sur puces » :

– la *simulation de processus physiques*, qui est à la base de la conception par ordinateur des objets et processus industriels modernes :

– *l'orchestration de services Web*, qui permet de réaliser des applications innovantes en composant les innombrables services disponibles sur la Toile, qu'il faut synchroniser et contrôler temporellement :

– enfin, *la composition et l'interprétation musicale*, en particulier pour les pièces mêlant des interprètes humains et électroniques.

J'ai insisté sur le fait que ces domaines apparemment très différents et socialement disjoints sont en fait techniquement proches, donc susceptibles des mêmes méthodes d'analyse et de développement. J'ai particulièrement apprécié de voir que les méthodes et langages synchrones (Esterel dans mon cas) ont séduit des compositeurs et informaticiens musicaux de haute volée, alors qu'elles avaient été conçues pour la programmation des systèmes temps-réel et la conception des circuits. Travailler avec les artistes est particulièrement gratifiant sur les plans techniques et humains.

---

3. http://www.college-de-france.fr/site/gerard-berry/course-2013-2014.htm [Ndlr].

4. http://www.college-de-france.fr/site/en-gerard-berry/course-2013-2014.htm [Ndlr].

5. La leçon inaugurale, *Le temps et les événements en informatique*, a été éditée sous forme imprimée (Fayard/Collège de France, 2013) et numérique (cf. http://books.openedition.org/cdf.3297). Elle est disponible en audio et vidéo sur le site Internet du Collège de France : http://www.college-de-france.fr/site/gerard-berry/inaugural-lecture-2013-03-28-18h00.htm [Ndlr].

J'ai ensuite évoqué la pratique quotidienne du temps et des événements en étudiant les expressions temporelles et événementielles de la langue de tous les jours, qui sont aussi adorablement fleuries que parfaitement imprécises. J'ai montré que les modélisations mathématiques classiques du temps en physique ou en automatique sont tout aussi insuffisantes pour les besoins de l'informatique, qui demande de travailler simultanément à diverses échelles de temps et divers niveaux d'abstraction. J'ai introduit les notions indispensables pour elle :

– *l'épaisseur de l'instant*, qui permet de voir des suites d'actions soit comme se déroulant en un temps fin dans un modèle d'exécution, soit comme atomiques et de durée conceptuellement nulle dans un modèle synchrone plus abstrait ;

– la distinction entre *modèles en temps continu*, standards en physique et en automatique, et *en temps discret*, plus naturels en informatique ;

– le *temps multiforme*, où l'on considère que la répétition de tout événement produit un quasi-temps analogue au temps physique : il n'y a pas de distinction à faire entre marcher dix minutes, dix kilomètres, ou dix mille pas, et il faut donc en parler de la même façon ;

– le *temps hiérarchique*, dans lequel les activités se décrivent à des degrés de temporalité emboîtés. Par exemple, en musique, on parle de la vibration interne de chaque note, de son ornementation, de l'organisation des notes en thèmes, de celle des retours et associations de thème, de celle des mouvements, etc. ;

– les *logiques et langages de programmation temporels*, qui permettent de parler des divers modèles de temps correspondants, et en particulier les langages synchrones – beaux produits de l'école française qui a toujours été en pointe dans ce sujet ;

– *l'échange espace-temps*, crucial en algorithmique et en conception de circuits électroniques. Il est impossible de concevoir des circuits performants sans comprendre à fond ce sujet aux techniques particulièrement élégantes ;

– la *sémantique mathématique* des modèles et langages temporels, indispensable pour les preuves de correction des raisonnements et des programmes.

J'ai développé ces idées à l'aide d'exemples que j'espère compréhensibles par tout un chacun. Ainsi, en relatant des travaux récents de collègues français[6], j'ai montré qu'une mauvaise relation sémantique entre modèles en temps discret et en temps continu produisait des aberrations dans les modeleurs mathématiques actuels et comment ces modèles pouvaient être corrigées à l'aide de techniques sémantiques originales.

Parler du temps, mais de manière formelle (cours 1 et 2)

Le premier cours du 2 avril 2013 a été double. La première heure a approfondi les notions générales présentées dans la leçon inaugurale, avec des exemples plus techniques. La deuxième heure a abordé l'étude des circuits combinatoires, en insistant sur leurs aspects temporels qui illustrent bien les différentes visions du temps.

---

6. Albert Benveniste, Timothy Bourke, Benoît Caillaud, Marc Pouzet, « Non-Standard Semantics of Hybrid Systems Modelers », *Journal of Computer and System Sciences (JCSS),* 78(3), 2012, 877-910.

Les formalisations du temps

Réaliser un système informatique, c'est prévoir ce qui peut se passer au cours du temps, le restreindre aux comportements souhaités, spécifier précisément et si possible formellement ces comportements, programmer pour que tout se passe comme spécifié, et vérifier la concordance de toutes ces actions. Les objectifs techniques sont de deux sortes, également difficiles : d'une part, spécifier, programmer et vérifier *a priori* la réalisation d'un système temporel et événementiel, et, d'autre part, analyser son exécution pour comprendre ses comportements dynamiques et être capable de trouver ses bugs.

Pour montrer que tout ne fonctionne pas toujours comme souhaité, j'ai présenté une collection de bugs liés au temps dans les systèmes embarqués grand public ou professionnels. Ils vont de problèmes idiots, comme l'absence de vérification d'une fonction temporelle apparemment non critique mais rendant l'objet inutilisable (lecteurs MP3 et jeux vidéo par exemple à cause des années bissextiles), à de plus subtils (erreurs cumulées d'arrondis dans la mesure du temps faisant échouer l'interception d'un missile Scud par un anti-missile Patriot lors de la guerre du Koweit, entraînant de nombreux morts).

J'ai ensuite analysé plus en profondeur les questions posées dans la leçon inaugurale, avec un *leitmotiv* : parler du temps, mais *de manière formelle*. J'ai expliqué par des exemples pourquoi les méthodes classiques sont insuffisantes.

Vibration et synchronisme dans les circuits combinatoires

La deuxième heure a été consacrée à la relation entre deux modèles fondamentaux, le modèle *vibratoire* en temps continu et le modèle *synchrone* en temps discret. J'ai pris comme illustration les *circuits combinatoires*, moteurs de tous les ordinateurs actuels. Les mêmes notions se retrouveront dans les cours suivants sur des exemples logiciels.

Considérons le circuit combinatoire *FullAdder* défini indifféremment par le dessin ou les équations de la figure 1, « oux » dénotant la fonction « ou exclusif ». Il possède trois bits d'entrées a, b et c, et calcule deux bits en sortie, leur somme s et leur retenue r. Il doit être vu de deux façons. Dans la vision *vibratoire*, ici électronique, il est constitué de fils et de portes construites avec des transistors. Si on applique des voltages binaires aux entrées, par exemple 0 V ou 1 V, les fronts électriques se propagent dans les fils à la vitesse de la lumière et les portes calculent dynamiquement le voltage de leur sortie en fonction de celui de leurs entrées. Ceci se réalise en temps fini et *prévisible* (minorable) par les électroniciens. Voici pourquoi. La propagation électrique se fait en parallèle dans le réseau. Chaque fil introduit un délai dépendant de son implantation sur la puce. Chaque porte introduit un délai dépendant du montage de ses transistors, de la température, et de ses valeurs d'entrées. Mais ces délais sont bornés et le circuit est acyclique : on peut donc calculer le plus long délai d'une entrée quelconque à une sortie quelconque, appelé *délai critique* ; les chemins correspondants sont appelés *chemins critiques*. Au plus tard après le délai critique, les voltages sont partout stables et les sorties valent les valeurs booléennes déterminées par les opérations logiques des portes en fonction des entrées. Le circuit se comporte comme une *machine vibratoire déterministe*, avec une propriété fondamentale de confluence : quel que soit l'ordre

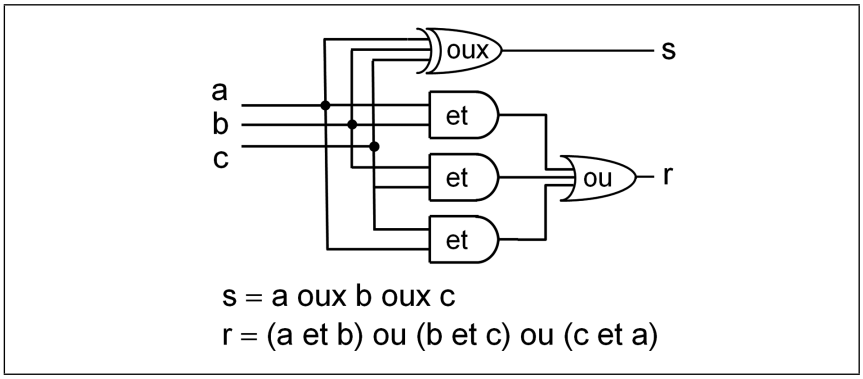

Figure 1 : Le circuit combinatoire *FullAdder*

de propagation des fronts, qui dépend de l'implantation et de la température, le résultat final est le même [7].

La *vision synchrone* utilise l'existence du délai critique pour s'abstraire de la propagation des fronts. Puisque les délais réels n'ont pas d'influence sur le résultat, on peut les ignorer et travailler en *temps discret*, en voyant le circuit comme un moyen parmi d'autres de résoudre les équations booléennes qui le décrivent. Le plus simple est d'adopter l'hypothèse de *synchronisme parfait*, qui considère que le circuit calcule les sorties en temps zéro. Alors, quand on compose des sous circuits, en séquence, en parallèle, ou par rebouclage de sortie sur des entrées (à condition que le graphe résultant reste acyclique), le calcul se fait toujours en temps zéro et toutes les simplifications booléennes sont applicables. Le concepteur du circuit peut travailler en synchronisme parfait, son système de CAO lui calculant le temps physique de stabilisation pour une technologie donnée. Bien sûr, il doit agir pour améliorer le temps réel de calcul si son circuit conceptuel n'est pas assez rapide par rapport aux besoins physiques.

Le reste du cours a approfondi la conception des circuits combinatoires, à travers l'addition de deux mots de n bits pour produire un résultat de n+1 bits. La méthode simple est celle de l'école : pour chaque couple de bits d'entrée a_i, b_i on instancie un additionneur $FullAdder_i$ d'entrées a_i, b_i et de retenue entrante c_i donnée par $c_0 = 0$ et $c_{i+1} = r_i$ pour $i > 0$. Les bits de la somme sont les sorties s_i des $FullAdder_i$ pour $0 \leq i < n$ et le bit de poids fort $s_n = r_{n-1}$. Le circuit est de taille linéaire en n, mais son délai critique déterminé par le chemin de retenues est aussi linéaire en n, ce qui est trop lent dès que n n'est pas tout petit.

Le délai critique s'améliore par divers é*changes temps-espace*. L'*additionneur de von Neumann* est fondé sur les idées clefs de *dichotomie* et de *spéculation*. L'idée de la dichotomie est de couper le mot d'entrée au milieu et de réaliser simultanément les additions des demi-mots de poids faibles et des demi-mots de poids forts. Mais la retenue entrante des poids forts est la retenue sortante de la somme des poids faibles. L'idée de la spéculation est de calculer simultanément les deux sommes

7. Mon cours de 2009-2010 a mentionné plusieurs propriétés de confluences dans des formalismes plus généraux, comme celle de Church-Rosser pour le lambda-calcul.

possibles des poids forts selon que leur retenue entrante est 0 ou 1, puis de sélectionner la bonne somme selon la retenue effective des poids faibles, l'autre somme étant purement et simplement ignorée. La sélection se fait à l'aide de *multiplexeurs* mux(x,y,z) calculant y si x vaut 1 et z si x vaut 0. Les sommes partielles des demi-mots de poids faibles et forts sont calculées récursivement par la même méthode. Le chemin critique est alors créé par l'arborescence de multiplexeurs qui résulte du développement de la récursion : il est logarithmique (de taille 6 au lieu de 64 pour 64 bits). Bien sûr, cette accélération temporelle a un coût en espace et en énergie, puisqu'on calcule des informations pour les ignorer ensuite.

Une autre méthode, inspirée par la programmation fonctionnelle, est due à J. Vuillemin et F. Preparata. Sa clef est que la fonction $f_i : r_i \rightarrow r_{i+1}$ de propagation de la retenue à l'étage i ne dépend que des deux bits a_i et b_i : c'est la fonction constante 0 si a_i et b_i valent tous deux 0, la fonction identité si l'un vaut 0 et l'autre 1, et la fonction constante 1 si les deux valent 1. Comme la retenue entrante globale est 0, on a $r_{i+1} = f_i(f_{i-1}(\ldots(f_1(f_0(0)))))$, donc $r_{i+1} = (f_i \circ f_{i-1} \circ \ldots \circ f_1 \circ f_0)(0)$. Puisque la composition de fonctions est associative, elle peut se calculer dichotomiquement, par exemple sous la forme $((f_7 \circ f_6) \circ (f_5 \circ f_4)) \circ ((f_3 \circ f_2) \circ (f_1 \circ f_0))$ pour n = 8. Les compositions peuvent donc être calculées par un arbre de calculs parallèles, donc en profondeur et temps logarithmiques. Dans le circuit, une fonction f : bool $\rightarrow$ bool est représentée par sa table de vérité, i.e. par la paire <f(0), f(1)> codée sur deux fils f_0 et f_1. L'application de f à x se calcule par $f(x) = mux(x, f_1, f_0)$, et la composition $g \circ f$, par $(g \circ f)_i = mux(f_i, g_1, g_0)$ pour i = 1,2. Voir les vidéos ou les planches du cours pour le circuit obtenu, particulièrement élégant.

Une addition bien différente utilise des *bases redondantes.* Une présentation très simple due à J. Vuillemin travaille avec des *nombres mous*, c'est-à-dire des paires de nombres A = (a', a'') codant de plusieurs façons la valeur a' + a''. Par exemple, 3 se code (0,3), (1,2), (2,1) ou (3,0). Il est facile de voir que l'additionneur de la figure 2 calcule en temps constant (indépendant du nombre de bits) un nombre mou dont la valeur est la somme des valeurs des nombres mous d'entrée. Bien sûr, on ne peut pas lire directement le résultat sans terminer par une autre addition sommant ses deux composantes, mais cette structure est très efficace quand on a une liste de nombres à additionner. On peut ainsi faire une multiplication efficace en construisant d'abord la matrice des produits individuels (simples conjonctions booléennes en base 2), en décalant les résultats comme à l'école, et en additionnant les nombres obtenus à l'aide d'un arbre dichotomique d'additionneurs mous : on obtient ainsi un résultat mou en temps logarithmique, qu'il ne reste plus qu'à durcir en un seul nombre à l'aide d'un dernier additionneur logarithmique, par exemple celui de von Neumann. Le résultat surprenant est que la multiplication se calcule effectivement en temps logarithmique. Les bases redondantes sont également fondamentales pour une division efficace, et c'est une erreur dans la tabulation de divisions molles dans une base non binaire qui a conduit au fameux bug du Pentium Pro qui a fait perdre 475 millions de dollars à Intel en 1994.

Le cours s'est terminé par le rappel d'un résultat important sur les circuits cycliques, déjà présenté en 2010. Il s'agit de généraliser la notion de chemin critique en se débarrassant de la contrainte d'acyclicité, donc de savoir quand un circuit cyclique calcule de façon déterministe et en temps prévisible pour tous délais des fils et portes. Cette question est importante d'une part car des circuits cycliques sont engendrés naturellement par synthèse à partir des langages de haut niveau et d'autre part car un circuit cyclique peut être exponentiellement plus petit que tout circuit

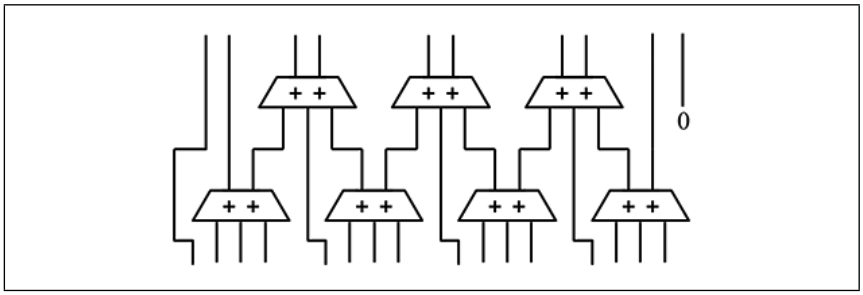

Figure 2 : Additionneur mou (sans retenues).

acyclique pour certaines fonctions booléennes [8]. Notre résultat est qu'un circuit cyclique est acceptable pour le critère précédent si et seulement si ses sorties peuvent être calculées à l'aide des seules lois de la *logique constructive*, qui interdit le tiers exclu « x ou non $x = 1$ ». Ainsi, le calcul électrique du circuit de *Hamlet* défini par l'équation *ToBe = ToBe or not ToBe* ne converge pas pour certains délais des fils et portes, car sa solution logique de l'équation demande le tiers exclu.

La restriction constructive est naturelle, car le tiers exclu demanderait une spéculation sur le résultat que les électrons sont bien incapables de réaliser. Mais la preuve du résultat est difficile. Une nouvelle version de cette preuve due à Mendler [9] sera présentée dans le cours de 2014. Elle fait intervenir une logique temporelle constructive mêlant étroitement aspects continus et discrets. Elle montre aussi qu'il faut voir un circuit non seulement comme une machine à calculer mais aussi comme une *machine à prouver*, ce qui donne une vue électrique de la correspondance fondamentale de Curry-Howard entre calculs en λ-calcul et preuves en logique intuitionniste [10].

Circuits synchrones et nombres 2-adiques (cours 3)

Les circuits séquentiels

Dans les *circuits synchrones*, les opérations se succèdent dans des étapes de temps discrètes. Un circuit synchrone se compose de deux parties conceptuellement distinctes, mais imbriquées dans l'implémentation physique : la partie *combinatoire*, sur laquelle repose le calcul à chaque instant, et les *registres*, qui mémorisent les valeurs d'un instant à l'autre. Les registres sont gouvernés par une *horloge*, qui produit des signaux carrés dont seuls les fronts montants nous intéressent ici. À chaque front montant, les registres échantillonnent et enregistrent simultanément les

8. S. Malik, « Analysis of cyclic combinational circuits », *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 13(7), 1994.

9. M. Mendler, T. Shiple et G. Berry, « Constructive Boolean Circuits and the Exactness of Timed Ternary Simulation », *Formal Methods in System Design*, Springer, 40(3), 2012, 283-329 : doi : 10.1007/s10703-012-0144-6.

10. Voir J.-Y. Girard, Y. Lafon et P. Taylor, *Proofs and Types*, Cambridge University Press (*Cambridge Tracts in Theoretical Computer Science*, 7).

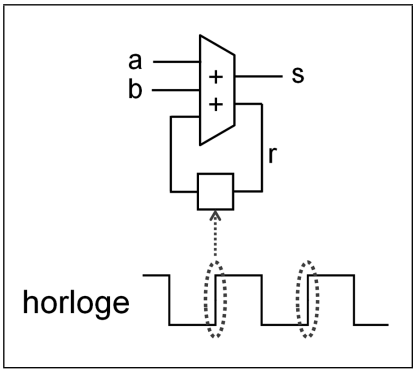

Figure 3 : L'additionneur sériel SerialAdder.

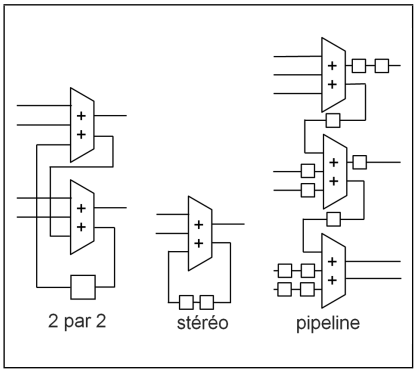

Figure 4 : Autres additionneurs sériels.

valeurs de leurs entrées [11], puis propagent ces valeurs sur leurs sorties pendant l'intégralité du cycle suivant. Un registre est donc un *décaleur* par rapport au temps discret. Pour simplifier, nous supposerons que les registres sont initialisés à 0 au premier cycle. Pour que le calcul se passe bien, il faut garantir que la distance temporelle entre fronts montants successifs de l'horloge est supérieure à la somme du délai critique de la partie combinatoire et du temps de basculement du registre (habituellement bien inférieur) : un circuit synchrone ne donc peut fonctionner à 1 gigahertz que si cette somme est inférieure à 1 nanoseconde. Dans les figures, un registre sera dessiné sous la forme d'un carré.

Les circuits synchrones permettent de travailler à la fois *dans le temps et dans l'espace*. L'exemple le plus simple est l'*additionneur sériel SerialAdder* de la figure 3. Les nombres d'entrée y sont présentés *poids faibles d'abord*, donc un 0 puis un 1 puis des 0 pour le nombre 2. Le *FullAdder* combinatoire fait une addition à chaque cycle, le registre propageant la retenue d'un cycle vers le suivant. Une paire de bits d'entrée produit un bit de somme à chaque cycle sur le fil de sortie du circuit. L'addition se fait dans le temps et non plus dans l'espace comme dans le cours précédent.

Trois autres circuits d'addition sérielle sont présentés dans la figure 4. Celui de gauche imbrique le temps et l'espace en traitant 2 bits à chaque cycle : la retenue des bits de rang pair est passée dans l'espace, celle des bits impairs dans le temps. Celui du milieu est plus subtil, car le doublement du registre entrelace dans le temps deux additions indépendantes, celle des nombres composés des bits d'ordre pair aux cycles pairs et celle des nombres composés des bits d'ordre impair aux cycles impairs. Nous l'appelons *additionneur stéréo* en référence au traitement entrelacé du canal droit et du canal gauche d'un enregistrement stéréophonique. Son principe se généralise à tout circuit séquentiel. Celui de droite utilise une technique fondamentale d'optimisation fondée sur l'échange espace-temps, le *pipeline*. Les opérations

11. À des micro-décalages temporels près dus aux petites différences de longueur des fils d'horloge les atteignant – la conception des arbres de distribution du signal d'horloge étant un sujet en soi.

combinatoires y sont coupées par des rangées de registres, ce qui accroît la vitesse d'horloge au prix d'une augmentation de la surface et de la *latence*, qui est le délai entre la fourniture des entrées aux circuits et celle des sorties par le circuit. C'est l'analogue de la méthode introduite par Henry Ford pour l'industrie automobile, les registres assurant le déplacement des objets dans la chaîne de montage.

Les nombres 2-adiques comme modèle des circuits séquentiels

Contrairement à l'additionneur combinatoire, l'additionneur sériel traite sans problème des chaînes de bits infinies, car la propagation de la retenue se fait dans le sens du temps. Or, les nombres écrits en base p avec une infinité de décimales mais poids faibles d'abord sont fondamentaux en mathématiques, où ils sont appelés *entiers p-adiques*. Nos suites infinies de bits sont donc des nombres 2-adiques. Les 2-adiques ont l'avantage d'unifier logique et arithmétique, car ils codent aussi les fonctions caractéristiques d'ensembles d'entiers en prenant comme éléments d'un ensemble les rangs des 1 dans la suite de bits du nombre 2-adique. Les extensions point à point des opérations booléennes $\sim$ (négation), $\wedge$ et $\vee$ calculent la complémentation, l'intersection et l'union des ensembles.

Notons les nombres 2-adiques en gras et leur développement binaire infini avec un indice 2 initial, le premier bit étant de rang 0. Le nombre **0**, qui dénote aussi l'ensemble vide, s'écrit $_2000\ldots$ Comme il est périodique, on l'écrit aussi avec la période entre parenthèse : $\mathbf{0} = {}_2(0) = \varnothing$. On a ensuite $\mathbf{1} = {}_210000\ldots = {}_21(0) = \{0\}$, $\mathbf{2} = {}_201000\ldots = {}_201(0) = \{1\}$, $\mathbf{3} = {}_211000\ldots = {}_211(0) = \{0,1\}$, etc. L'addition se fait de gauche à droite en propageant la retenue à l'infini, donc dans le sens de l'écriture. Elle est réalisée par le circuit *SerialAdder* de la figure 3. La multiplication se fait aussi par prolongation à l'infini de la multiplication classique. Ces deux opérations donnent aux nombres 2-adiques une structure d'anneau (nous n'utiliserons pas ici la structure de corps 2-adique).

Les opposés se calculent par « complémentation à 2 infinie » : $-\mathbf{1} = {}_211111\ldots = {}_2(1) = \{0,1,2,\ldots\}$, puisque $\mathbf{1}+(-\mathbf{1}) = \mathbf{0}$, puis $-\mathbf{2} = {}_201111\ldots = {}_20(1) = \{1,2,3,\ldots\}$, etc. Les nombres périodiques correspondent aux rationnels à dénominateur impair. Par exemple, pour $\mathbf{x} = {}_2101010\ldots = {}_2(10) = \{i \mid i \text{ pair}\}$, on peut écrire $\mathbf{x} = {}_2100000\ldots + {}_2001010$, donc $\mathbf{x} = \mathbf{1} + \mathbf{4x}$ puisque ajouter 00 en tête d'un 2-adique revient à le multiplier par 4, d'où $\mathbf{x} = -\mathbf{1/3}$.

Une relation 2-adique fondamentale est la relation *arithmético-logique* $\forall \mathbf{x}.\ \mathbf{x} + \sim\mathbf{x} = -\mathbf{1}$, évidente car l'addition de **x** et $\sim$**x** se fait sans retenue. Si on pose $\mathbf{y} = \sim(-\mathbf{1/3}) = {}_2010101\ldots = {}_2(01) = \{i \mid i \text{ impair}\}$, il est facile de voir qu'on a $\mathbf{y} = -\mathbf{2/3}$ puisque $-\mathbf{1/3} + \mathbf{y} = -\mathbf{1}$ d'après l'identité arithmético-logique. Je laisse le soin au lecteur de vérifier que tout nombre rationnel –p/q possède bien un développement 2-adique, ce qui n'est pas vrai pour les rationnels à dénominateur pair : 1/2 n'est pas représentable car son premier bit b devrait vérifier b+b = 1, ce qui est impossible.

Le remarquable travail de Jean Vuillemin [12] établit les rapports entre les 2-adiques et les circuits. Les points essentiels sont résumés dans la figure 5. D'abord, le circuit *FullAdder* trouve en 2-adiques une définition bien plus intéressante que sa définition

12. J. Vuillemin, « On circuits and numbers », *IEEE Trans. on Computers*, 43(8), 1994, 868-79.

purement combinatoire de la figure 1. Si on note **a**, **b**, **c**, **s, r**, les suites d'entrées et sorties de l'additionneur au cours du temps vues comme 2-adiques, alors *FullAdder* vérifie l'invariant 2-adique **a + b + c** = **s + 2r**, qui remplace avantageusement le calcul booléen par le calcul numérique. Le second point est qu'un registre est un multiplieur par **2** pour les 2-adiques, puisqu'il sort un 0 initial puis les bits d'entrée retardés. Si on rajoute une négation booléenne de chaque côté du registre, représentée par un cercle, le « registre à oreilles » obtenu calcule **1+2x** pour l'entrée **x**.

Cette formalisation rend triviales des preuves qui seraient difficiles en suivant les bits un par un. Par exemple, il est trivial que le circuit de gauche dans la figure 6 calcule **z** = **x**–**y** : l'invariant de *FullAdder* donne **x + ~y + c** = **z + 2r**, et celui du registre à oreilles donne **c** = **1+2r**. Donc **x + ~y + 1** = **z** en simplifiant par **2r**, et **x**–**y** = **z** puisque **~y + 1** = –**y** d'après l'identité arithmético-logique. La preuve est triviale, mais elle ne dit vraiment rien de ce qui se passe au niveau des bits !

Le cours a présenté de superbes circuits pour calculer **3x** et **x/3**, généralisables à tout **p** impair, la quasi-division **1/1-2x**, et, en point d'orgue la quasi-racine carrée **√(1–8x)**, dont la preuve de 5 lignes est étourdissante. Je renvoie aux vidéos et transparents pour les détails.

Les entiers 2-adiques forment également un espace topologique compact avec la *norme 2-adique* $\| \mathbf{x} \| = 2^{-n}$ si $n = min\{i \mid \mathbf{x}_i = \mathbf{1}\}$. La distance associée est définie par $d(\mathbf{x},\mathbf{y}) = \| \mathbf{x}–\mathbf{y} \|$. Autrement dit, deux entiers 2-adiques **x** et **y** sont à distance 2^{-n} si et seulement si le segment initial commun de leur développement est de longueur n. J. Vuillemin a démontré le théorème fondamental suivant :

Théorème : une fonction **f** : 2z → 2z est calculable par un circuit synchrone (éventuellement de mémoire infinie) si et seulement si elle est contractante pour la distance 2-adique : $\forall \mathbf{x}, \mathbf{y}.\ d(\mathbf{f}(\mathbf{x}),\mathbf{f}(\mathbf{y})) \leq d(\mathbf{x},\mathbf{y})$.

Toute fonction calculée par un circuit synchrone est trivialement contractante, car l'inégalité exprime que les sorties sont égales sur les n premiers bits si les entrées sont égales sur les n premiers bits. La réciproque, bien plus subtile, construit une forme normale dite *Sequential Decision Diagram* (SDD), fondée sur une représentation dans le temps et dans l'espace de la table de vérité de la fonction.

Un résultat constamment utilisé mais rarement prouvé est qu'on peut toujours reboucler toute sortie sur toute entrée si tout chemin d'une entrée vers une sortie est coupé par un registre. Ce n'est en fait qu'une conséquence du théorème de Banach disant que toute fonction Lipschitzienne sur un compact a un point fixe unique : dire que tout chemin est coupé revient à dire $d(\mathbf{f}(\mathbf{x}),\mathbf{f}(\mathbf{y})) < d(\mathbf{x},\mathbf{y})$, équivalent à $d(\mathbf{f}(\mathbf{x}),\mathbf{f}(\mathbf{y})) < 0{,}6\ d(\mathbf{x},\mathbf{y})$ d'après la définition de la distance, CQFD.

Le cours a présenté d'autres propriétés fondamentales de l'interprétation 2-adique : le fait qu'on puisse implémenter les fonctions *continues* et pas seulement synchrones en travaillant avec des pairs de fils validité-valeur, donc en rendant le temps *élastique* comme expliqué plus loin ; le fait qu'on puisse linéariser la forme normale SDD de Vuillemin en un nombre 2-adique qui représente la table de vérité dans le temps et l'espace de la fonction ; le résultat consistant à voir ce nombre comme définissant une série formelle, qui est algébrique si et seulement si la fonction peut être implémentée par un circuit de mémoire finie ; le fait extraordinaire que, vu comme un nombre réel en le préfixant par « 0, », ce nombre est soit rationnel, soit transcendant, etc. Beaucoup de travail reste à faire dans ce domaine magnifique mais trop peu connu à mon goût.

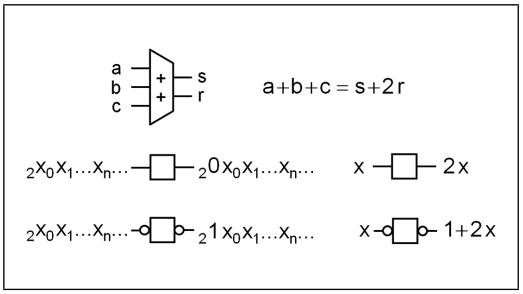

Figure 5 : Les circuits vus sous l'angle 2-adique.

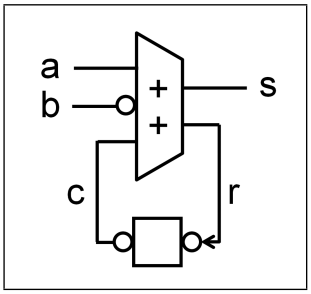

Figure 6 : Soustracteur sériel 2-adique.

Séminaire: l'évolution des microprocesseurs (Olivier Temam)

Olivier Temam, directeur de recherches à l'Inria a montré pourquoi les performances des microprocesseurs ont évolué de façon considérable depuis l'introduction de l'Intel 4004 en 1971, tandis que restait inchangé leur principe de base issu de l'architecture de von Neumann des années 1940 : se relier à une mémoire externe, et exécuter un programme enregistré en mémoire sur des données elles aussi en aussi mémoire en décodant ses instructions et en exécutant les calculs à l'aide d'unités arithmétiques et logiques et de mémoires locales temporaires.

L'amélioration des performances est venue de deux sources : la décroissante de la taille des fils et transistors, conduisant à une augmentation exponentielle du nombre de transistors par puce (loi de Moore) et à l'augmentation des vitesses d'horloge, et l'amélioration de l'architecture logique des processeurs fondée sur le parallélisme à grain fin et sur plusieurs échanges espace/temps : pipeline, spéculation, etc. Enfin, les mémoires devenant de plus en plus lentes par rapport aux processeurs, plusieurs niveaux de mémoire-cache gérés par des algorithmes sophistiqués accélèrent le temps moyen d'accès aux données.

O. Temam a expliqué que le défi actuel est celui de l'énergie dépensée. Comme on ne peut plus accélérer les horloges sans brûler les circuits, la seule solution pour améliorer les performances est de mettre plusieurs processeurs sur chaque puce (actuellement 8, bientôt des centaines ou des milliers), ce qui améliore aussi la tolérance aux erreurs de fabrication puisqu'on peut débrancher les processeurs non fonctionnels au test. Mais un nouveau problème apparaît, celui du « silicium noir » : tout processeur qui travaille chauffe, et il faut vite l'arrêter et faire migrer son programme ailleurs pour qu'il se refroidisse – une bonne partie des processeurs étant simplement en train de se reposer ! L'architecture matérielle est donc un sujet en évolution constante, d'autant plus que de nouvelles formes de pensée émergent avec les processeurs quantiques, biomimétiques, etc. Elles sont pleines de promesses, mais il ne faut pas confondre ses envies et la réalité, comme on le voit trop souvent dans des affirmations pseudo-scientifiques présomptueuses.

Le langage Esterel v5 (cours 4 et 5)

Le langage synchrone Esterel v5 a été développé à partir de 1982 dans mon équipe mixte École des mines/Inria à Sophia-Antipolis. En parallèle, celle de N. Halbwachs

et P. Caspi développait Lustre à Grenoble et celle d'A. Benveniste et P. Le Guernic développait Signal à Rennes. Esterel visait les applications à contrôle dominant (grands automates, protocoles de communication, interfaces homme-machine, etc.) alors que Lustre et Signal visaient les applications de type flot de données (contrôle continu, traitement du signal). Les trois équipes, chacune composée d'informaticiens et d'automaticiens, ont convergé vers l'utilisation du principe synchrone avec un temps logique nul et un temps physique prédictible, implémenté par une boucle à 4 temps attente / lecture des entrées / calcul de la réaction / production des sorties. L'approche synchrone conduit à un parallélisme déterministe et à la prédictibilité des spécifications et de l'implémentation. Les sémantiques formelles des trois langages ont dirigé leur conception et leur implémentation.

En 1990, en collaboration avec l'équipe de J. Vuillemin chez Digital Equipment, nous avons compris que notre hypothèse de synchronisme logiciel rejoignait celle faite implicitement par les concepteurs de circuits. Cela nous a permis de résoudre deux problèmes cruciaux pour Esterel : la compréhension de la causalité synchrone et l'explosion exponentielle en taille du code généré dans l'implémentation par des automates finis déterministes. Elle a aussi permis de mieux comprendre la correspondance entre automates finis déterministes et non-déterministes [13]. Enfin, traduire les programmes en circuits a permis d'effectuer des vérifications formelles automatiques de propriétés de sûreté : l'ascenseur ne peut pas voyager la porte ouverte, l'avion ne peut pas rétracter son train d'atterrissage quand il est au sol, etc. Le compilateur Esterel a été couplé avec le vérifieur formel TiGeR de Madre, Coudert et Touati développé au même labo qui tirait parti des nouveautés sur la représentation de fonctions booléennes par les BDD (*Binary Decision Diagrams*). Ces vérifications feront l'objet de plusieurs cours en 2014-2015.

Design et sémantique d'Esterel v5 (cours 4)

Ce cours a présenté les fondements d'Esterel : l'hypothèse synchrone, les instructions temporelles primitives ou dérivées, le style de programmation induit [14], la réduction potentiellement exponentielle de la taille des sources par rapport aux méthodes classiques d'automates et la sémantique constructive du langage [15] fondée sur une logique constructive de réécriture des termes. L'idée de cette sémantique est directement issue de l'utilisation de la logique booléenne constructive pour les circuits cycliques.

Compilation d'Esterel v5 (cours 5)

Le cours suivant a présenté la compilation d'Esterel en logiciel ou matériel. Historiquement, elle a connu plusieurs phases : traduction des programmes en automates déterministes par calcul symbolique (avec L. Cosserat), traduction en

13. Détails dans G. Berry, « Penser, modéliser et maîtriser le calcul informatique », *Cours et travaux du Collège de France*, 110e année, Paris, Collège de France, 897-926.

14. Voir le programme du coureur à pieds dans la leçon inaugurale, sous forme vidéo, ou dans *Le temps et les événements en informatique,* G. Berry, Fayard/Collège de France, 2013 : en ligne : http://books.openedition.org/cdf/3297.

15. *The Constructive Semantics of Pure Esterel*, http://www-sop.inria.fr/members/Gerard.Berry/Papers/EsterelConstructiveBook.pdf

automates bien plus efficace avec meilleure gestion de la causalité (avec G. Gonthier, base de la première industrialisation), puis traduction en circuits fondée sur la sémantique constructive qui évite toute explosion du code. Cette traduction peut être vue comme la construction pour chaque programme d'un *réseau de preuves* regroupant l'ensemble des preuves constructives du programme pour toutes les entrées possibles, en rejoignant encore une fois le paradigme de Curry-Howard, « calculer c'est prouver ».

Pour les cibles logicielles, les sémantiques mathématiques et les implémentations fondées sur elles ont été déterminantes pour les succès initiaux d'Esterel et pour son industrialisation par CISI-Ingénierie, Simulog, puis ILOG. Elles ont contribué au dialogue avec les utilisateurs, en particulier en avionique (Dassault Aviation). Esterel v5 a été utilisé en protocoles de communication (Bell Labs, ATT, British Telecom), en robotique (Inria). Pour les circuits, Esterel v5 a été commercialisé par Synopsys comme composant d'un logiciel de CAO de systèmes électroniques : il a été directement utilisé par Cadence Design Systems pour la conception conjointe matériel/logiciel, et par Intel et Texas Instruments pour la spécification de circuits de contrôle.

Séminaire (Emmanuel Ledinot)

E. Ledinot est directeur des études scientifique amont de Dassault Aviation. Avec son équipe, il a été un pionnier de l'utilisation des méthodes formelles, et particulièrement d'Esterel et de ses outils de vérification formelle. Esterel n'aurait pas pu atteindre la maturité sans les exemples qu'il nous a fournis, de taille et de complexité bien plus grandes que ceux que nous imaginions au laboratoire.

E. Ledinot a présenté l'évolution des systèmes informatiques pour l'avionique, qui ont grandi en taille d'un facteur 100 entre 1980 (Mirage F1) et 2000 (Rafale 2000), avec une avionique de plus en plus modulaire. Il a détaillé les problèmes de séquencement et de parallélisation des fonctions, puis montré la difficulté de les réaliser avec les méthodes de programmation déterministe classiques, qui demandent de tout ordonnancer à la main. Il a expliqué pourquoi Esterel résout ce problème en assurant automatiquement un ordonnancement correct par construction et en engendrant un code aux performances prévisibles. Il a montré pourquoi la composition de sous-systèmes développés par des personnes différentes conduit à des problèmes de causalité complexes, résolus dans les dernières versions du compilateur Esterel v5 fondées sur la sémantique constructive. Il a enfin montré pourquoi le nouveau système SCADE 6 décrit ci-dessous est un outil essentiel pour la complexité encore supérieure des applications futures, car il unifie contrôle continu et contrôle discret dans un formalisme graphique familier aux ingénieurs et mathématiquement bien défini.

Séminaire : SCADE 6 (Bruno Pagano)

Bruno Pagano est directeur scientifique d'Esterel Technologies. Il est responsable du développement de SCADE 6, le nouveau langage synchrone textuel/graphique distribué par cette société et appliqué à grand échelle en avionique, en ferroviaire, en industrie lourde, et, espérons le, bientôt dans l'automobile.

SCADE 6 est un harmonieux mélange de Lustre, base de SCADE 4 utilisé par exemple pour l'Airbus A380, d'Esterel pour sa vision du contrôle synchrone, et des SyncCharts de C. André qui une version synchrone des Statecharts de D. Harel.

SCADE 6 y ajoute un traitement fonctionnel des tableaux de données et de processus. Il est intégré dans un atelier logiciel, qui comporte des interfaces avec des méthodes de spécification de haut niveau (Simulink, SysML, etc.) et des outils de génie logiciel (documentation, simulation, gestion de tests, etc.). La compilation de SCADE 6 est soumise à des contraintes dures, le compilateur étant lui-même certifié au niveau maximal A de la norme DO-178B. Sa certification repose sur une documentation et des tests extensifs, et, techniquement, sur un mécanisme de traçabilité complète du code source au code objet. Elle est facilitée par la sémantique mathématique du langage et lui donne un avantage commercial considérable.

B. Pagano a présenté le langage et ses méthodes de compilation, en insistant sur les points techniques originaux : traduction du graphique en texte, ordonnancement conduisant à des codes générés efficaces, détection des problèmes de causalité par typage statique, polymorphisme des types et traitement des tableaux – très propre conceptuellement mais dont l'efficacité reste améliorable.

Esterel v7, spécification et synthèse de circuits (cours 6 et 7)

La conception d'Esterel v7 a commencé en 1999 au *Strategic CAD Lab* d'Intel à Portland (États-Unis), en commun avec Michael Kishinevsky, mon autre maître à penser dans le domaine des circuits avec J. Vuillemin. M. Kishinevsky avait fait des expériences concluantes avec Esterel v5, trouvant son parallélisme synchrone et ses instructions temporelles bien adaptées au besoin, et ayant vérifié l'efficacité des algorithmes de synthèse et de vérification. Mais il lui semblait indispensable d'augmenter la puissance descriptive du langage en lui intégrant la spécification des chemins de données : représentation des nombres, structuration multiple des mots, tableaux de données et de processus, etc. En 2000 a été créée la société Esterel Technologies, ayant pour but d'industrialiser Esterel v7 et de le commercialiser dans les domaines de CAO de circuits et de logiciels temps-réels critiques. J'en ai été le directeur scientifique jusqu'en 2009. Le double objectif a été abandonné lors du rachat de SCADE en 2003 : Esterel v7 a été affecté aux circuits et SCADE aux logiciels, le travail sur Esterel v7 restant séminal pour la définition de SCADE 6.

Mon équipe s'est engagée dans plusieurs années de développement du langage, du compilateur et de son environnement de développement Esterel Studio, accompagné d'expérimentations en recherche ou applications préindustrielles chez Intel, Texas Instruments (TI), ST microelectronics et NXP (ex Philips). Esterel Studio est ensuite entré en production chez TI et ST : contrôles de mémoires variés, dont les difficiles DMAs (*Direct Memory Access*) pour les applications voraces en données, gestion de bus, de réseaux sur puce et de communications, contrôle de systèmes vidéo, etc.

Le langage a été cédé en 2007 à l'IEEE [16], en vue de sa standardisation, conduite par un groupe de chercheurs et d'industriels. Sa rigueur formelle a permis que les circuits soient développés plus rapidement, avec moins d'ingénieurs, et surtout de vérifieurs, tout en conservant une excellente efficacité. Ce langage a permis aux architectes et micro-architectes de tester et de modifier rapidement leurs idées en utilisant nos simulateurs graphiques et outils de vérification formelle, tout en étant confiants dans la qualité de la synthèse finale. Je montrerai en 2014 à quel point

16. Institute of Electrical and Electronics Engineers, projet P1178.

l'industrie nous a posé des problèmes scientifiques plus difficiles que ceux que nous connaissions en recherche, et je présenterai ensuite les méthodes de vérification formelle pour Esterel v7.

L'aventure s'est terminée en mars 2009 à la suite de la crise de 2008, comme pour beaucoup de produits innovants à cette époque. La standardisation IEEE du langage, presque terminée, a été suspendue. Esterel Studio a été racheté par Synfora, lui-même racheté par Synopsys, leader du domaine. Synopsys a décidé de mettre Esterel Studio au congélateur, ce qui fait que même ses auteurs ne peuvent plus l'utiliser, alors que tout l'outillage était auparavant accessible gratuitement pour l'enseignement et la recherche.

Esterel v7, le design du langage (cours 6)

Les instructions temporelles d'Esterel v7 sont issues de celles d'Esterel v5, mais simplifient l'émission simultanée ou conditionnelle de signaux ; la structure modulaire a été améliorée et rendue plus générique. Les signaux d'Esterel v5 ont également été étendus pour répondre aux besoins des circuits ; leurs valeurs et bits de présence sont tous deux optionnels, et les valeurs peuvent êtres temporaires, mémorisées, ou retardées. Mais la nouveauté principale est le traitement des données et des tableaux. Esterel v7 introduit des types numériques bien plus précis que ceux des langages habituels, comme le type unsigned<M> qui dénote l'ensemble des M entiers naturels i tels que $0 \leq i < M$ et le type signed<M> qui dénote l'ensemble des 2M entiers signés i tels que $-M \leq i < M$. Il y a plusieurs raisons pour introduire ces types. D'abord, en circuits, le raisonnement sur la taille en nombre de bits de l'écriture binaire est insuffisant : par exemple, on a les typages 4 : unsigned<5>, 5 : unsigned<6> et 4*5 = 20 : unsigned<21>. Si 4 et 5 s'écrivent avec 3 bits en binaire, leur produit 20 s'écrit avec 5 bits, et non les 6 qu'on obtient en additionnant leurs tailles individuelles ; ceci devient évident grâce au type unsigned<21>. Ensuite, le binaire n'est pas le seul codage utile en matériel, où l'on travaille aussi en code Gray, un seul bit changeant à chaque incrémentation, ou en code *onehot*, avec un seul bit à 1 dans l'écriture du nombre. Les types signed et unsigned sont abstraits et indépendants de leurs représentations, qui peuvent être de nombre de bits variable. Enfin, les unsigned servent à indexer les tableaux et notre typage rend facile de tester les débordements d'index à la compilation par analyse de types ou vérification formelle.

En Esterel v7, les tableaux sont de types et de dimensions quelconques, qui peuvent être paramétriques. Ils s'appliquent aux signaux et aux valeurs. Une instruction de réplication statique permet de construire des tableaux de processus pour manipuler les tableaux de données et de signaux. Enfin, Esterel v7 permet deux types de manipulations d'horloge : le *masquage de cycle* (*clock-gating*), qui possède une vraie sémantique mathématique, au contraire des langages classiques Verilog/VHDL, et l'utilisation modulaire d'*horloges multiples*. Ces aspects seront détaillés en 2014.

Le cours s'est terminé par la présentation d'un mini-filtre vidéo qui illustre les différents aspects du langage, ainsi que la façon de l'utiliser pour « pipeliner » facilement les circuits et donc diminuer leur temps de cycle.

La compilation d'Esterel v7 (cours 7)

La compilation d'Esterel v7 est délicate. Pour la compilation en circuits, le principe de base est resté celui d'Esterel v5, mais la technique a été compliquée par quatre facteurs : le traitement des tableaux, la compilation modulaire de gros programmes, la gestion des horloges, et, hélas, la mauvaise qualité sémantique des langages cibles VHDL et Verilog. La clef du succès final est l'*optimisation du contrôle*, qui s'appuie sur des algorithmes développés originellement pour Esterel v5 et améliorés pour Esterel v7. L'idée est de calculer symboliquement l'espace des états mémoire de contrôle atteignables pour les entrées légales, puis de simplifier la logique et de réallouer la mémoire de contrôle en fonction de ces états atteignables. Cette optimisation utilise le logiciel TiGeR, mentionné plus haut pour la vérification formelle d'Esterel v5 [17]. Les résultats en taille et vitesse de circuits sont comparables à ceux des *designs* manuels, et souvent meilleurs.

La compilation en logiciel C ou C++ est devenue indispensable pour deux raisons. D'abord, c'est le moyen le plus rapide de tester le circuit en simulation ; ensuite, les ingénieurs logiciels doivent développer les applications avant même que le circuit n'existe physiquement, pour respecter le *time-to-market*. Esterel v7 a repris des techniques développées ailleurs pour Esterel v5, en particulier par S. Edwards à l'université Columbia (États-Unis) et D. Potop à l'Ecole des mines [18]. Ces techniques sont complexes et le cours n'en a présenté que le principe. Le cours s'est terminé par la démonstration de l'animation des sources lors de la simulation, fondée sur des techniques de traçabilité du code source dans le code généré.

Séminaires : circuits asynchrones et élastiques (Jordi Cortadella)

Jordi Cortadella, professeur à Universitat Politècnica de Catalunya à Barcelone, est spécialiste de la conception et de la synthèse des circuits électroniques. Il s'intéresse aux alternatives aux circuits synchrones classiques, qui cherchent à éviter leurs inconvénients : la distribution de l'horloge à grande échelle est difficile et coûteuse en énergie ; il faut couper les fils trop longs par des registres, ce qui complique beaucoup le contrôle ; enfin, les puces comportent souvent plusieurs horloges indépendantes, avec des problèmes de communication entre zones asynchrones non solubles par les méthodes synchrones.

Les circuits asynchrones

J. Cortadella a d'abord présenté les circuits *asynchrones*. L'idée est de remplacer la synchronisation globale que réalise l'horloge par une synchronisation locale et conjointe des données et du contrôle. Il y en a deux variantes, décrites brièvement ici.

Le *dual-rail encoding* code toute donnée sur deux fils, les paires de valeurs <0,1>, <1,0> et <0,0> codant respectivement les valeurs 0, 1 et *espace*, la paire <1,1> étant inutilisée. Toute suite de deux valeurs doit être séparée par un espace. Une porte spéciale dite *Müller C-element* permet de synchroniser les calculs parallèles. Le *dual rail* garantit un fonctionnement logique indépendant des délais

17. Esterel Studio utilise un autre système plus général de vérification, Prover SL de Prover Technologies.

18. Voir D. Potop, S. Edwards et G. Berry, *Compiling Esterel*, Springer, 2007.

des fils et portes : il est plus rapide en moyenne que le synchrone, car on peut signaler la fin du calcul dès qu'elle est advenue au lieu d'attendre toujours le pire cas. On peut ainsi additionner des nombres de n bits en temps moyen $\log_2(\log_2(n))$ au lieu de $\log_2(n)$. Mais ce gain se fait au prix d'un doublement de la surface et de l'énergie, ce qui limite cette technique à des cas spécifiques.

La seconde méthode associe deux fils de contrôle à chaque opérateur, un fil *valid* allant de l'émetteur au récepteur pour signaler la validité de la sortie et un fil *stop* allant du récepteur à l'émetteur pour signaler l'impossibilité d'accepter une nouvelle entrée. Pour assurer la stabilisation du résultat, le fil *valid* traverse un délai explicite supérieur au délai critique de l'opérateur ; le fil *stop* est alors remis à 0 pour signaler à l'émetteur l'attente d'une nouvelle donnée. Le circuit n'a plus d'horloge globale, mais son analyse conduit aux mêmes contraintes de pire cas que pour un circuit synchrone. Mais un texte ne remplacera jamais les superbes animations de J. Cortadella, et j'engage vivement le lecteur à regarder les vidéos.

Malgré leur intérêt, les circuits asynchrones n'ont pas le même succès industriel que les circuits synchrones, pour trois raisons : leur conception assistée par ordinateur est plus complexe et moins développée que celle des circuits synchrones ; leur test après fabrication peut être aussi plus complexe ; enfin, l'industrie de la conception des circuits est très conservatrice pour des raisons que je discuterai en 2014.

Les circuits élastiques

Les circuits *élastiques* généralisent les notions précédentes d'asynchronisme d'une façon qui s'applique aussi aux circuits synchrones. L'idée est de donner davantage de liberté architecturale en dissociant le temps logique du calcul du temps physique de la réalisation. Un exemple classique est celui des circuits GALS (*Globally Asynchronous Locally Synchronous*), où des îlots localement synchrones sont connectés par des synchroniseurs multi-horloges ou des files d'attente asynchrones. Le problème délicat est alors celui de la *métastabilité des registres* (je renvoie à mon cours de 2013-2014).

Un bel exemple plus récent est celui des *circuits synchrones élastiques.* Ils restent synchrones, mais utilisent le protocole *valid/stop* des circuits asynchrones. L'horloge est conservée et la conception logique reste la même qu'en synchrone, mais les registres sont automatiquement remplacés par des paires de *verrous transparents* (*transparent latches*) qui stockent leur entrée pendant une phase de l'horloge et se comportent comme de simples fils pendant l'autre phase. Les fils *valid/stop* sont gérés par une couche asynchrone qui commande des portes de *masquage d'horloge* (*clock gating*) inhibant les transitions des verrous transparents. L'élasticité vis-à-vis de l'horloge est alors garantie. On peut couper tout fil trop long par une *bulle* (registre non initialisé) sans modifier le contrôle du circuit, ce qui est impossible en design synchrone, et on peut réaliser des pipelines plus élaborés. Ces nouveaux circuits représentent un progrès conceptuel certain, mais le conformisme industriel fait que leur succès est loin d'être garanti.

HOP et HipHop, l'orchestration d'activités Web (cours 8)

Ce cours a été directement couplé au séminaire, les deux orateurs intervenant à tour de rôle. Il s'agit d'une recherche en pleine activité sur la programmation et l'orchestration d'activités Web. Manuel Serrano, directeur de recherches Inria, a

présenté le langage et le système HOP [19], qui permettent de remplacer la tour de Babel habituelle des langages du Web par un seul langage fonctionnel hérité du langage Scheme : écriture du serveur, codage HTML du client et communication client-serveur sont ainsi unifiés. Un programme HOP combine dans un seul code source les actions des clients et du serveur, distinguées par de simples annotations ; le compilateur HOP les répartit automatiquement et engendre l'ensemble des communications nécessaires à la cohérence du tout. Sur le serveur, HOP est compilé et exécuté nativement, avec une excellente efficacité. Sur les clients, HOP est compilé en Javascript, ce qui assure une portabilité optimale au prix de quelques limitations.

HOP permet de construire des applications Web bien plus simplement qu'avec les méthodes classiques, soit directement, soit en composant les innombrables services Web déjà existants (un service s'appelle comme une page Web, mais il rend des données au lieu d'afficher directement une page). Mais coordonner l'appel, le retour et les messages d'erreurs des services asynchrones distants est un problème complexe. Pour le traiter, nous avons défini et implémenté le langage HipHop, une version d'Esterel intégrée dans HOP, qui permet de tirer parti de toute la puissance du parallélisme synchrone et des instructions temporelles d'Esterel pour réaliser l'orchestration des services. Au contraire d'Esterel v5 et v7 qui laissaient à l'utilisateur le soin d'intégrer le code généré dans leur environnement propre, HipHop traite ce problème en profitant de la puissance intégrative de HOP due à son caractère fonctionnel d'ordre supérieur [20]. Il introduit une notion de *machine réactive* réalisant l'interface entre les événements asynchrones de HOP et la boucle d'interaction synchrone de HipHop [21]. Il permet aussi de développer des applications GALS (*Globally Asynchronous Locally Synchronou*s) en faisant coopérer de façon asynchrone plusieurs machines réactives synchrones.

Tout cela a été illustré par l'exemple d'un système de recherche et de jeu de morceaux musicaux qui fait appel à différents sites de téléchargement de musique et de métadonnées associées. Il est indispensable de regarder les animations des vidéos pour en savoir plus.

Séminaires: le temps en informatique musicale

Le cours 2013-2014 s'est terminé par deux séminaires sur le temps en musique, sujet qui me tient particulièrement à cœur. Outre son intérêt artistique, la musique rejoint la plupart des problèmes précités : épaisseur de l'instant, temps multiforme, temps continu / temps discret, etc. Il se trouve que les techniques synchrones détaillées dans cet article s'adaptent bien à la création et à l'interprétation musicale, qu'elles sont déjà utilisées dans des outils d'informatique musicale, et qu'elles pourraient se révéler cruciales pour un problème largement ouvert, celui des langages d'écriture de la musique contemporaine. Les rencontres déterminantes

19. http://hop.inria.fr.

20. Scheme est une extension du λ-calcul, cf. cours de 2009-2010.

21. Sujet déjà exploré dans le cadre d'Esterel, en particulier par F. Boussinot et L. Mandel avec l'intégration de la programmation réactive dans les langages classiques, et par D. Gaffé à Nice pour l'interfaçage du code généré par Esterel avec les systèmes d'exploitation standards.

pour moi ont été celles d'Arshia Cont lors de mon cours de 2009-2010, puis celle de Philippe Manoury lors de la première de sa composition *Tensio* en novembre 2010. Nous interagissons depuis sur l'ensemble des questions liées au temps.

La musique du temps réel (Philippe Manoury)

Philippe Manoury, pionnier de la musique mixte hommes/informatique, est compositeur de musique contemporaine, professeur au conservatoire de Strasbourg et professeur émérite à l'université de San Diego. Sa recherche artistique et technique implique des réflexions constantes sur le rôle du temps en musique.

Aux débuts de la musique électronique, les sons étaient enregistrés puis reproduits dans un tempo prédéfini. Ceci contrastait avec le fait que les musiciens jouent en interaction constante entre eux et avec la partition, communiquant par le regard autant que par le son, et gérant leur tempo en fonction de plusieurs paramètres : choix expressifs, acoustique du lieu, etc. Pour laisser plus de champ aux interprètes, K-H. Stockhausen pouvait insérer des séquences de liberté instrumentale entre les parties électroniques. P. Manoury souhaitait aller beaucoup plus loin en associant l'électronique à l'intimité de la musique par une véritable interaction entre interprètes et sons de synthèse. Ceci a été rendu possible par l'usage des ordinateurs, qu'il voit comme assurant trois fonctions : la synthèse de séquences sonores prédéterminées, la possibilité d'engendrer des séquences aléatoires et la possibilité de comprendre dynamiquement l'environnement et de s'y adapter grâce à des capteurs adéquats. Un progrès important a été le *suivi de partition*, qui asservit le tempo de la musique électronique à celui des interprètes par analyse des sons qu'ils produisent ; il a été développé par Miller Puckette [22] et utilisé pour la première fois dans la pièce *Jupiter* de Manoury. Antescofo, présenté ci-dessous, a approfondi cette approche par une analyse fine des variations de tempo essentielles pour l'expression musicale.

À travers plusieurs exemples, P. Manoury a insisté sur la multiplicité des temps présents en musique et sur les rapports non-triviaux entre temps abstrait de la partition et temps concret de l'interprétation. Par exemple, dans les contrepoints savants de J-S. Bach, plusieurs phrases peuvent se dérouler simultanément ou successivement selon des bases de temps différentes ; leurs partitions sont de nature purement logique, omettant toute indication de tempo physique de phrasé, ou d'ornementation. En analysant des fragments de certaines de ses œuvres récentes (*Tensio*, pour quatuors à corde humain et électronique, *Echo-Daimónon*, concerto pour piano), P. Manoury a illustré l'utilisation des temps multiples dans ses compositions, et insisté sur les possibilités offertes par l'informatique : phrases impossibles à jouer par l'homme, développements fondés sur des suites aléatoires (musique markovienne), déclenchement ou arrêt de sons continus ou de séquences de sons par des événements temporels ou déduits du jeu des interprètes, adaptation des sons de synthèse selon l'analyse des formants d'une voix chantée, etc. Voir et écouter la vidéo s'impose.

P. Manoury a insisté sur l'importance de l'écriture musicale et de son niveau d'abstraction pour la création de musiques complexes. Les premières écritures à base de tablatures n'étaient pas transposables d'un instrument à un autre, alors que l'écriture musicale actuelle est largement indépendante des instruments. Mais il faut

22. Développeur des systèmes de synthèse sonore interactive Max/MSP, maintenant industriel, et Pure Data.

aller beaucoup plus loin dans la précision et l'abstraction, car les sous-entendus des partitions classiques ne sont pas accessibles aux ordinateurs : les compositeurs développent souvent leurs propres notations, et une vraie notion de partition algorithmique générique reste à développer.

P. Manoury a enfin insisté sur les risques engendrés par l'industrialisation des outils musicaux quant à la pérennité des œuvres actuelles. Puisqu'il n'est pas sûr que les outils utilisés pour la composition d'une œuvre soient maintenus par l'industrie comme le sont les violons et pianos, il n'est pas non plus sûr que les pièces actuelles pourront être réinterprétées à l'avenir, contrairement à celles de la musique savante précédente.

Antescofo (Arshia Cont)

Arshia Cont, chercheur à l'IRCAM[23], développe le système Antescofo pour l'interprétation de pièces mixtes homme/électronique, qui amplifie l'idée que l'électronique doit être finement dirigée en temps réel par les hommes. Après avoir présenté les problèmes informatiques liés à l'interaction homme/ordinateur, il a expliqué le cœur d'Antescofo, qui est un ensemble d'algorithmes qui assurent le suivi adaptatif et anticipatif de partitions en comprenant en temps réel comment les musiciens humains organisent et varient leur tempo et leur phrasé pour donner la musicalité souhaitée à l'œuvre, puis adaptent finement le jeu électronique à ce tempo dynamique. Ces algorithmes évolués sont issus du traitement probabiliste du signal et des neurosciences de la perception des sons et du temps, avec une modélisation hiérarchique des divers temps en jeu, du temps logiques de la partition aux micro- et macro-temps continus ou discrets de l'interprétation en passant par le temps propre de l'ordinateur. Les sons électroniques déclenchés par Antescofo sont calculés par le système Max/MSP, le standard du domaine, soit par traitement classique de signal, soit par modélisation physique d'instruments. Le déclenchement de ces sons informatiques est commandé par un langage algorithmique directement fondé sur nos idées de langages synchrones, qui permet d'associer aux événements temporels (notes, durées) et de contrôler des phrases musicales informatiques complexes. Un point difficile est le rattrapage des erreurs inévitables des instrumentistes ou du logiciel de suivi. Ce rattrapage d'erreurs est indispensable car, quoi qu'il arrive, *the show must go on*. Tout cela a été illustré en temps réel par des exemples et un dialogue entre le clarinettiste Jérôme Comte (Ensemble inter-contemporain) et l'ordinateur.

Antescofo est utilisé dans de nombreux concerts de musique contemporaine à travers le monde (dont toutes les œuvres récentes de P. Manoury et d'autres musiciens comme M. Stroppa), et très apprécié des compositeurs. Ce sera un laboratoire idéal pour les futurs travaux sur les partitions algorithmiques, dont le langage actuel d'Antescofo peut être vu comme un premier embryon.

23. Institut de recherche en coordination et acoustique musicale.

CONFÉRENCES À L'EXTÉRIEUR

05.09.2012, trois conférences chez Thalès Air Systems, Rungis, « Pourquoi le monde devient numérique : les racines scientifiques et techniques », « Les inversions mentales de l'informatique » et « Le bug, la maladie du certain ».

06.09.2012, aux élèves de la promotion entrante de l'École centrale, « Pourquoi le monde devient numérique : les racines scientifiques et techniques ».

07.09.2012, au colloque « Art, harmonie, cerveau » de Mouans-Sartoux (06), « Le rôle du temps et de l'espace en harmonie musicale ».

27.09.2012, au 20e anniversaire du laboratoire Verimag, Grenoble , « Synchronous/ Asynchronous Web Orchestration with Hop and HipHop ».

04.10.2012, lors du déjeuner des membres fondateurs du cercle des amis d'Universciences à la Cité des sciences et de l'industrie, « Les inversions mentales de l'informatique ».

24.10.2012, au colloquium de l'université Pierre et Marie Curie , « Time and Events in Informatics ».

25.10.2012, au laboratoire Programmes, preuves et systèmes de l'université Paris-Diderot, « Time and Events in Informatics ».

17-20.11.2012, au workshop Synchron au Croisic, « The Informatics of Time and Events ».

04.12.2012, au département d'informatique de l'ENS Paris, « Le temps et les événements en informatique ».

10-14.12.2012, à l'Académie des sciences de Pékin, « Time and Events in Informatics » ; à l'université Tsinghua, « Revisiting Finite-State Systems ».

22.01.2013, orateur invité à la séance publique de l'Académie des sciences « Les enfants et les écrans ».

08.02.013, à l'association Art, sciences, pensée de Mouans-Sartoux (06), « L'informatique: de la révolution technique à la révolution mentale ».

21.02.2013, à l'Assemblée nationale : participation à l'audition publique « le risque numérique », organisée par l'Office parlementaire d'évaluation des choix scientifiques et technologiques.

28.02.2013, au CLAS (Comité local d'action sociale) du Collège de France, « L'informatique: de la révolution technique à la révolution mentale ».

11.03.2013, à Sciences Po Paris, « Le bug informatique ou la maladie du certain ».

21.03.2013, à la journée « La nature est-elle un grand livre ? » au Fresnoy, Lille, table ronde sur le livre Walden de Henry David Thoreau.

25.03.2013, aux 20e anniversaire du laboratoire Programmes, preuves et systèmes de l'université Paris-Diderot, « Le temps et les événements en informatique ».

27.03.2013, participation au débat organisé par Cédric Villani après la projection du film Codebreaker sur Alan Turing, cinéma Grand Action, Paris.

24.04.2013, aux Journées informatique fondamentale de Paris Diderot, « Playing with Time and Space in Circuits and Programs ».

27.05.2013, dans le cycle Philosophie de l'informatique à École normale supérieure de Paris, « Parler du temps, du joli vague de la langue à la précision de l'informatique ».

03.06.2013, à la soirée de l'association des Alumni de l'université de Stanford, « Why and How the World Becomes Digital », maison France-Amérique, Paris.

AUTRES INTERVENTIONS

Émissions de radio

J'ai participé aux émissions de radio suivantes :
– *L'éloge du savoir*, France culture, interview diffusée le 1er avril 2013.
– *Autour de la question*, Radio France internationale, 18 avril 2013.
– *7 milliards de voisins*, Radio France internationale, 28 juin 2013.
– *Éducation/Jeunesse*, France Info, 24 mai 2013, à propos du rapport de l'Académie de sciences sur l'enseignement de l'informatique.

Activités liées à l'enseignement de l'informatique

En 2012-2013, j'ai continué l'action sur la place de l'informatique dans l'enseignement général initiée en 2007-2008. Après trois ans de formation de professeurs volontaires, l'enseignement de spécialité en terminale scientifique « Informatique et sciences numériques » a été mis en place à la rentrée 2012 comme premier enseignement de science informatique dans notre pays.

Le gouvernement a décidé d'étendre cet enseignement à toutes les terminales scientifiques en 2014. Dans sa « feuille de route sur le numérique », il a également mentionné que des efforts considérables devraient être faits pour que l'enseignement général utilise les outils numériques, et qu'une réflexion devrait être menée sur l'enseignement de la science informatique à tous les niveaux. La communauté informatique salue cette prise de conscience, mais la considère comme insuffisante par rapport aux besoins : il faut intégrer *dès maintenant* la science informatique dans l'enseignement général depuis le plus jeune âge. Dans ce domaine devenu essentiel pour l'économie, la culture et la démocratie, l'ignorance est encore reine dans quasiment tous les pans de la société, ce qui nous place en simples consommateurs de ce qui se conçoit et se fabrique ailleurs. Pour que notre pays retrouve son rôle de créateur du monde de demain, qui sera encore plus numérique et donc informatique, il est indispensable que l'ensemble de la population soit formé de façon adéquate pour comprendre au lieu de subir les évolutions rapides de l'informatique et profiter de tous les emplois qu'elle crée directement ou indirectement.

Plusieurs rapports internationaux se sont unanimement exprimés sur ce sujet. J'ai participé à un rapport européen intitulé *Informatics education: Europe cannot afford to miss the boat*[24] , publié en avril 2013 par l'association *Informatics Europe* qui regroupe des universités et des centres de recherche et de formation européens et par la section européenne de *l'Association of Computing Machinery* (ACM), la principale société savante en informatique au plan mondial.

J'ai coordonné le rapport de l'Académie des sciences intitulé « L'enseignement de l'informatique : il est urgent de ne plus attendre[25] », auquel a aussi participé Serge Abiteboul, titulaire de la chaire Informatique et sciences numériques en 2011-2012. J'ai présenté ce rapport à l'ensemble des chefs d'établissement de l'Académie de Rouen le 23 mai 2013, et au conseil scientifique de la DEGESCO (direction

24. http://germany.acm.org/upload/pdf/ACMandIEreport.pdf.
25. Mai 2013, http://www.academie-sciences.fr/activite/rapport/rads_0513.pdf.

générale de l'enseignement scolaire) le 20 juin 2013. Plusieurs émissions de radio lui ont été consacrées.

Autres activités

Je suis membre du Conseil d'administration de l'Agence nationale de la recherche et du Conseil scientifique de l'université Pierre et Marie Curie.

J'ai présidé le comité d'experts scientifiques mandaté par l'Institut de sûreté et de radioprotection nucléaire (IRSN) sur la sûreté des logiciels critiques dans les centrales nucléaires (rapport paru en novembre 2013).

J'ai organisé au Collège un séminaire régulier sur le temps en informatique, avec des participants de l'UPMC, de Paris-Diderot, de l'IRCAM, d'Inria et du CEA.

J'ai participé au numéro de la *Revue du Grand Paris* « La ville doit être souple [26] », consacré à l'impact de l'informatique sur la ville et les transports du futur. Je tiens une chronique scientifique régulière dans le journal « Les Échos ».

J'ai participé aux jurys des prix et grand prix de l'Académie des sciences, au jury du grands prix Milner de la Royal Society à Londres et au jury des bourses l'Oréal-UNESCO pour les femmes et la science.

PUBLICATIONS

Les publications ci-mentionnées sont antérieures à l'arrivée du professeur au Collège de France.

M. Serrano et G. Berry, « Multi-tier programming in Hop », *Communications of the ACM*, 5(8), août 2012 (http://queue.acm.org/detail.cfm?id=2330089).

M. Mendler, T. Shiple et G. Berry, « Constructive boolean circuits and the exactness of timed ternary simulation », *Formal Methods in System Design*, Springer, 40(3), 283-329, 2012 : doi : 10.1007/s10703-012-0144-6 (http://link.springer.com/article/10.1007%2Fs10703-012-0144-6).

26. http://www.veolia.fr/france-veolia/ressources/files/1/43073,revue-grand-paris-decembre-2012.PDF