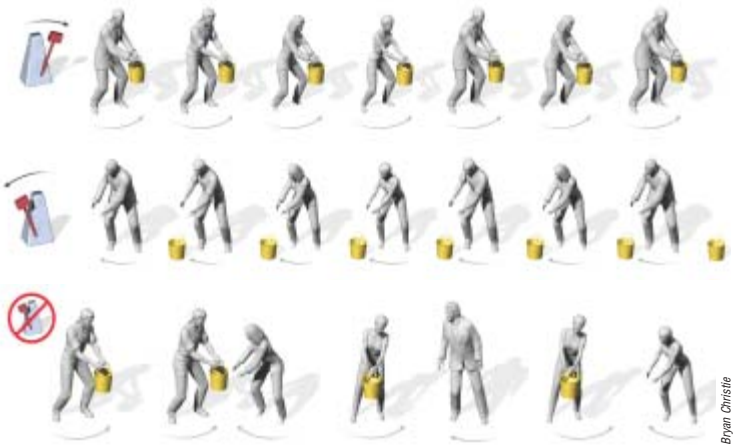

Bryan Christie

2. LES CHAÎNES DE PASSEURS DE SEAUX : on peut utiliser cette métaphore pour décrire le flux des données dans un ordinateur. Un système informatique synchrone est comparable à une chaîne de passeurs où chaque personne reçoit et passe les seaux au rythme du tic-tac d'une horloge. Quand l'horloge fait «tic», chaque passeur avance son seau vers la personne suivante *(en haut)*. Quand l'horloge fait «tac», chaque passeur attrape le seau posé par la personne précédente *(au milieu)*. Au contraire, un système asynchrone ressemble à une chaîne de passeurs ordinaire : chaque personne tenant un seau peut le passer à la suivante dès qu'elle a les mains libres *(en bas)*. Le désordre engendré par cette perte de cohésion complique certes la conception des puces, mais le flux de données traitées augmente : la vitesse de la chaîne n'est plus limitée par le circuit le plus lent.

fréquence est un multiple de la fréquence originelle). De tels signaux peuvent interférer avec les téléphones portables, la télévision et les systèmes de navigation des avions qui emploient les mêmes fréquences. Comme les systèmes asynchrones n'ont pas de cadence fixe, ils répartissent leur énergie de rayonnement sur l'ensemble du spectre radio, de sorte que l'émission dans chacune des fréquences est faible.

La conception asynchrone présente un avantage supplémentaire : elle permet la construction de ponts entre des ordinateurs de fréquences différentes. Les réseaux d'ordinateurs hétérogènes – on les nomme des grappes – qui prolifèrent depuis quelques années, mettent en relation des ordinateurs rapides avec des machines plus lentes, qui, sinon, auraient été mises au rebut. Ces grappes permettent de traiter des problèmes complexes en répartissant les tâches de calcul entre les différentes machines. Un tel système est par essence asynchrone : les divers constituants ont des cadences différentes. Le transfert de données contrôlées par une horloge vers une machine contrôlée par une autre horloge nécessite des ponts asynchrones, car les données peuvent être, au cours des calculs, «désynchronisées» par rapport à l'horloge de l'ordinateur qui reçoit l'information.

En dernier lieu, bien que la conception asynchrone présente des difficultés, elle offre une grande flexibilité. Étant donné que les circuits d'un système asynchrone ne sont pas tenus de suivre tous le même rythme, les concepteurs ont plus de liberté pour choisir quelles parties du système participeront au calcul, et pour décider comment elles interagiront. En outre, le remplacement de n'importe quelle partie par une version plus rapide augmente la vitesse de l'ensemble du système, alors que dans un système cadencé, l'augmentation de la vitesse nécessite généralement la mise à jour de tous les constituants.

Afin de décrire le fonctionnement des systèmes asynchrones, nous utilisons souvent la métaphore d'une chaîne de passeurs de seaux. Un système cadencé est comparable à une chaîne de passeurs de seaux où chaque participant doit passer et recevoir les seaux au rythme d'une horloge. Au premier «tic», chacun pose son seau devant son voisin de gauche, par exemple ; quand l'horloge fait «tac», chacun attrape le seau déposé par le voisin *(voir la figure 2)*. Le rythme de cette chaîne est défini par le temps que met la personne la plus lente de la chaîne à déplacer le seau le plus lourd. Même si les seaux sont légers, tout le monde doit attendre le battement de l'horloge avant de passer le seau suivant.

Coopération locale

Dans une chaîne de passeurs de seaux asynchrones, c'est la coopération locale qui détermine le rythme, et non une horloge commune. Chaque personne tenant un seau peut le passer à la suivante dès que cette dernière a les mains libres. Quand les seaux sont légers, ils font rapidement le trajet d'un bout à l'autre de la chaîne. De surcroît, quand il n'y a pas d'eau à transporter, tout le monde peut se reposer en attendant. Si une personne lente ralentit l'ensemble de la chaîne, on la remplace et l'ensemble retrouve des performances optimales.

Dans les ordinateurs, les chaînes de passeurs de seaux sont nommées des «pipelines». Un pipeline ordinaire comporte environ une demi-douzaine de relais, ayant chacun un rôle analogue à celui d'un des passeurs de seaux. Au lieu de manipuler des seaux d'eau, chaque relais effectue l'une des tâches nécessaires à l'exécution de l'instruction. Par exemple, un processeur exécutant l'instruction $C = A + B$ doit aller chercher la signification de l'instruction «+» dans la mémoire, récupérer les nombres correspondant à A et à B, en faire la somme et stocker le résultat dans la mémoire C.

Un pipeline cadencé effectue ces opérations à un rythme ne dépendant ni de l'opération à effectuer ni de la taille des nombres. Additionner *1* et *1* demande autant de temps qu'additionner deux nombres de 30 chiffres. En revanche, dans un pipeline asynchrone, la durée de chaque action dépend de l'opération effectuée, de la taille des nombres ainsi que de l'emplacement des données dans la mémoire (de la même manière que dans la chaîne de passeurs, la quantité d'eau dans un seau peut influer sur le temps qu'il faut pour le passer). Sans horloge pour réguler ses actions, un circuit asynchrone recourt à des circuits de coordination locale. Ces circuits échangent des signaux de fin d'exécution afin de s'assurer qu'à chaque étape les actions ne commencent que lorsque le circuit dispose de toutes les données nécessaires. Les deux circuits de coordination les plus importants sont nommés le *Rendez-vous* et l'*Arbitre*.

Un élément *Rendez-vous* indique le moment où toutes les données requises par un circuit pour effectuer une tâche sont parvenues à destination. Par exemple, un circuit de division arithmétique doit disposer à la fois du dividende (par exemple 16) et du diviseur (par exemple 2) avant de pouvoir diviser l'un par l'autre (et donner le résultat 8).

Un type de circuit *Rendez-vous* est nommé élément C de Muller, du nom de David Muller, qui fut professeur à l'Université de l'Illinois. Un élément C de Muller est un circuit logique avec deux entrées et une sortie *(voir l'encadré page 39)*. Quand les deux entrées d'un élément C de Muller sont VRAI, sa sortie devient VRAI. Quand les deux entrées

sont FAUX, la sortie devient FAUX. Dans les autres cas, la sortie ne change pas. Les éléments C de Muller servent notamment à contrôler le flux des données. Par un agencement astucieux de plusieurs d'entre eux, ils permettent de gérer l'avancée de données le long d'une chaîne de passeurs électronique sans le besoin d'une horloge.

Notre groupe de recherche a récemment introduit un nouveau type de circuit *Rendez-vous*, nommé GasP *(voir l'encadré page 41)*. Le GasP est dérivé d'une famille plus ancienne de circuits mise au point par Charles Molnar, qui travailla aux Laboratoires *Sun Microsystems*. Ch. Molnar avait baptisé sa création asP* (pour protocole de pulsation asynchrone et symétrique). Nous y avons ajouté le *G*, pour obtenir l'onomatopée *(gasp)* qui, en anglais, traduit la surprise, surprise créée ici par la rapidité des nouveaux circuits. Nous avons établi que les modules GasP ont une rapidité et une efficacité énergétique aussi bonnes que les éléments C de Muller, qu'ils s'assemblent mieux avec les circuits ordinaires et qu'ils sont beaucoup plus faciles à utiliser dans des circuits complexes.

Les circuits *Arbitre* remplissent, quant à eux, une autre fonction essentielle pour les ordinateurs asynchrones. Un *Arbitre*, tel un agent de la circulation posté à une intersection qui décide quel véhicule passera le premier, orchestre le traitement des données. S'il n'y a qu'une requête, l'*Arbitre* donne immédiatement le feu vert à l'action correspondante, et met en attente une éventuelle seconde requête jusqu'à ce que la première soit terminée. Quand l'*Arbitre* reçoit deux requêtes quasi simultanément, il doit décider à laquelle accorder la priorité. Par exemple, quand deux processeurs réclament l'accès à une mémoire au même moment, le rôle de l'*Arbitre* est de n'accorder l'accès qu'à un processeur à la fois. L'*Arbitre* garantit que deux actions ne se déroulent jamais en même temps.

L'âne de Buridan

Bien que les circuits *Arbitre* n'accèdent jamais à plus d'une requête à la fois, il n'existe aucun *Arbitre* qui tranche entre les diverses requêtes en un temps limité. Les *Arbitre* actuels prennent leurs décisions très rapidement en moyenne, généralement en quelques picosecondes (un millième de milliardième de seconde). Cependant, quand les demandes sont très rapprochées, les circuits mettent parfois deux fois plus longtemps à trancher, voire, dans quelques rares cas, dix fois plus.

Les difficultés auxquelles est confronté l'*Arbitre* pour prendre ses décisions rappellent celles de l'âne de Buridan. Selon cette parabole attribuée à Buridan, philosophe français du XIV^{e} siècle, un âne affamé et assoiffé, se trouvant à égale distance d'un picotin d'avoine et d'un seau d'eau, mourrait de faim et de soif parce qu'il serait incapable de choisir entre l'un ou l'autre. Nous sommes confrontés tous les jours à ce type de dilemmes, par exemple lorsque deux

Fonctionnement d'un circuit *Rendez-vous*

Les circuits *Rendez-vous* gèrent le flux des données à travers des circuits sans horloge centrale. Ici un pipeline électronique est contrôlé par une chaîne de circuits *Rendez-vous*, nommés éléments C de Muller. Chacun de ces éléments autorise le passage des données *(en orange)* seulement quand l'étage précédent est «plein» (les données sont prêtes à poursuivre leur chemin) et que l'étage suivant est «vide».

Chaque élément C de Muller a deux entrées et une sortie. La sortie devient FAUX quand les deux entrées sont FAUX, et redevient VRAI quand les deux entrées sont VRAI (dans le diagramme, les signaux VRAI sont en bleu, et les signaux FAUX en rouge). L'inverseur fait en sorte que les entrées initiales diffèrent de sorte qu'au départ, tous les étages sont vides.

Supposons que l'entrée de gauche soit VRAI et celle de droite FAUX *(a)*

Un changement de l'entrée de gauche de VRAI à FAUX *(b)* indique que l'étage de gauche est plein, c'est-à-dire que des données sont arrivées. Comme les deux entrées de l'élément sont maintenant toutes deux FAUX, l'élément bascule sa sortie sur FAUX. Cette modification a trois conséquences : des données sont transmises dans le pipeline pour poursuivre leur trajet (les portes s'ouvrent), un signal FAUX est renvoyé vers l'élément C de Muller précédent pour que l'étage soit vidé et un signal FAUX est expédié vers l'élément C de Muller suivant, de sorte que l'étage suivant est plein *(c)*.