HAL
open science

Le temps du calcul et le temps des ordinateurs

Philippe Matherat

To cite this version:

Philippe Matherat. Le temps du calcul et le temps des ordinateurs. 2006. hal-00089817

HAL Id: hal-00089817

https://hal.science/hal-00089817v1

Preprint submitted on 23 Aug 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ccsd-00089817, version 1 - 23 Aug 2006

Le temps du calcul et le temps des ordinateurs

Philippe Matherat
GET - Télécom-Paris - Comelec / CNRS - LTCI (UMR 5141)
philippe.matherat@telecom-paris.fr

Résumé

S'interroger sur l'implémentation physique du calcul distribué conduit à s'interroger sur les modèles de temps, temps des modèles de calcul du côté logique, et temps des théories mécaniques du côté physique. Pour un électronicien, *implémenter* revient à *traduire*, depuis un langage logique vers le langage de la physique, c'est-à-dire à *simuler*. Nous suggérons que ce travail est proche de celui du logicien qui compare les puissances d'expression des langages distribués, et proche aussi de celui du physicien qui cherche à écrire les lois de la nature en langage mathématique. Dans les trois cas, l'interrogation sur la structure du temps est centrale. Ce texte est une transcription de l'exposé donné à l'ENST, le 27 avril 2006.

1 Introduction

Notre titre évoque deux sortes de temps :

- Le temps mathématique du calcul,
- Le temps physique des ordinateurs.

Un ordinateur a pour fonction d'effectuer des calculs dans le monde physique, et doit donc concilier ces deux sortes de temps. Il s'agit d'*utiliser le temps physique pour simuler le temps du calcul*. Tout l'art de l'ingénieur en électronique est de comprendre en quoi ces deux temps éventuellement différents permettent ou non de *simuler* l'un par l'autre.

Simuler est ici un mot central. Il apparaît dans des contextes apparemment éloignés :

- Dans le contexte de comparaison des langages logiques quand on parle de pouvoir d'expression des langages,
- Dans le contexte du physicien lorsqu'il effectue des calculs afin de comparer les prédictions d'une théorie à des résultats expérimentaux,

– Dans le contexte de l'électronique quand on parle de simulation d'une machine logique par une machine matérielle (*réalisation* d'un ordinateur).

Ce n'est que cette troisième activité qui est appelée "*art de l'électronicien*", mais il nous semble qu'il n'y a peut-être pas de raison de la distinguer des deux autres. Nous allons explorer cette question de la réalisation des ordinateurs, en nous laissant guider par la problématique de la *synchronisation*. Nous terminerons par la présentation des circuits asynchrones DI, un des modèles de calcul distribué dont le modèle de temps est proche du modèle de temps de la physique relativiste.

Outre cette question de *synchronisation*, nous évoquerons la question de *dissipation du calcul* qui a été pour nous une étape importante dans la prise en compte de ces questions sur le temps.

2 Logique synchrone et physique classique

2.1 Circuits synchrones et automates

Quand on apprend à réaliser un ordinateur synchrone, on s'aperçoit assez rapidement que tout ce qui concerne les opérations combinatoires et arithmétiques est très simple par rapport à ce qui concerne le temps et la synchronisation. Si on associe la valeur booléenne $Vrai$ (ou 1) à la proposition physique "*le courant passe*", on peut traduire simplement les opérations booléennes (figure 1).

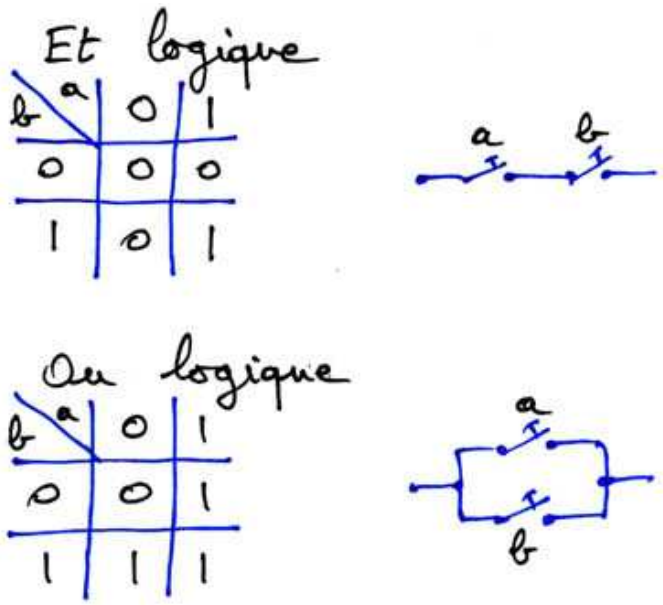

Fig. 1 – Fonctions booléennes

Un article de Shannon de 1938 [1] montre que l'algèbre de Boole est équivalente à l'algèbre des interrupteurs associés en série ou en parallèle. Comme il est aisé de traduire les opérations arithmétiques en opérations booléennes, on peut faire tous les calculs ainsi, et c'est sur ce principe que sont construits les ordinateurs.

Mais les choses se compliquent lorsqu'on se voit contraint d'introduire la mémorisation d'une variable booléenne afin d'enchaîner les calculs. On utilise par exemple ce qu'on appelle une *bascule D* (figure 2).

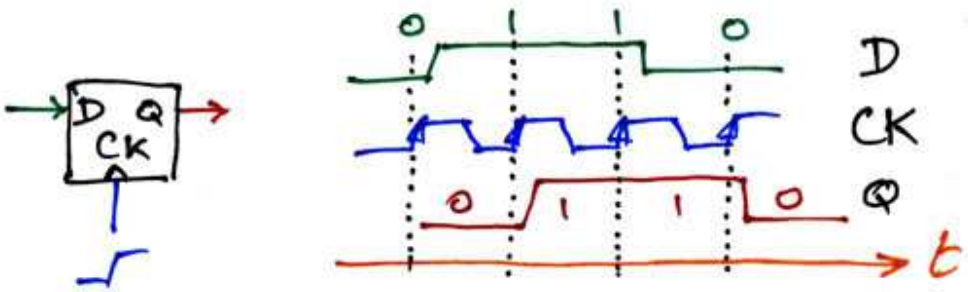

Fig. 2 – Une bascule D

Il y a une entrée D, une sortie Q et une entrée d'horloge CK. Lorsqu'on impose un front montant sur CK, cela échantillonne D, cette valeur est enregistrée à l'intérieur de la bascule et apparaît sur sa sortie Q.

Il y a apparition d'un *lien causal* entre les valeurs de D et Q *à des instants différents*, séparés par un front d'horloge.

Pour décrire cela mathématiquement ou logiquement, il ne suffit plus d'une table de vérité, il faut un autre objet, l'*automate*, qui peut être représenté par un graphe (figure 3).

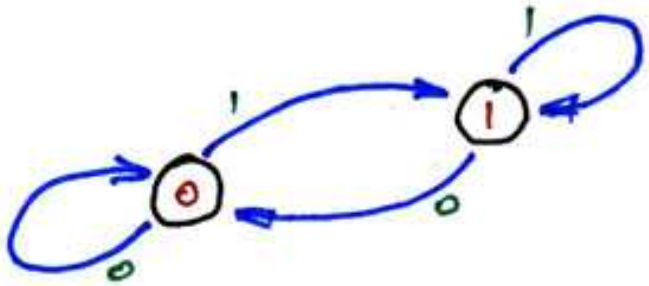

Fig. 3 – Graphe de la bascule D

Les cercles représentent les deux *états* possibles, 0 et 1, de la mémoire. À chaque coup d'horloge (front montant), on effectue une *transition*, représentée par une flèche, étiquetée par la valeur de la donnée. La donnée est vue ici comme une cause externe qui opère un choix interne, le choix de la transition qui sera effectuée au moment du prochain front d'horloge.

S'il y a plusieurs points mémoire avec la *même horloge*, il y a plus de deux états et les calculs peuvent être variés (figure 4). (Dans cette figure, nous n'avons pas représenté les valeurs des données.)

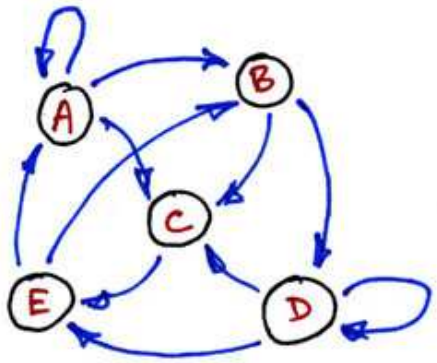

FIG. 4 – Graphe d'automate

- Du côté mathématique, ceci débouche sur la *théorie des automates*, qui avec la *machine de Turing*, permet de modéliser tous les calculs (calculs séquentiels, finis ou infinis) [2] [3].
- Du côté des électroniciens, il s'agit de ce qu'on appelle la *logique synchrone* car il y a une horloge commune globale. On parle de "*machine à états*".

Le temps est ici *un nombre entier*, c'est-à-dire à la fois :
- *un* nombre, donc *le même partout* (dans tout l'espace du circuit électronique).
- un nombre *entier*, donc un *temps discret* (les périodes d'horloge sont indexables par un entier).

Nous allons voir que c'est très différent du temps de la physique, qui n'est ni "le même partout" ni "discret". Pourtant, les électroniciens savent très bien réaliser ce genre de montages. Ce qui les rend possibles, c'est que l'objet mémoire et le signal d'horloge sont les éléments qui prennent en charge la *synchronisation*. Il s'agit de *simuler* le temps discret et global du modèle logique, à l'aide du temps physique.

2.2 La notion d'*état* et la dissipation du calcul

Que signifie le mot "*état*", largement utilisé tant en physique (état d'un système mécanique ou thermodynamique) qu'en logique (état d'un automate fini) ?

Il faut entendre : un état est une *classe d'équivalence de passés*. Prenons un exemple pour illustrer cela (figure 5).

Ce montage fait la somme de tous les bits passés, en série, et ne garde que le bit de poids le plus faible. Il ne mémorise donc que la parité de la somme. Si nous ne connaissons que S, la sortie de la mémoire, nous ne connaissons qu'un bit du passé. Pourtant ce bit est fonction de

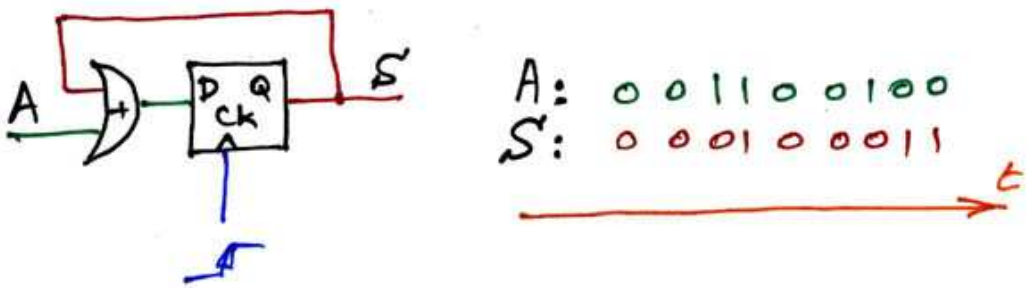

Fig. 5 – Calcul de parité

toute l'histoire passée ! Ce montage crée une partition de tous les passés possibles en deux classes d'équivalence. C'est la *classe d'équivalence* qui est appelée l'*état présent*.

On ne connaît pas le passé, on ne connaît que le présent des mémoires !

Un calcul, c'est un parcours dans un graphe d'automate. En général, c'est irréversible, à cause des convergences (figue 6).

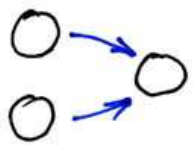

Fig. 6 – Convergence dans un graphe

Quand on effectue une transition vers un état atteint par une convergence, on perd de l'information sur le passé, c'est-à-dire qu'on ne pourra pas faire le chemin à l'envers. C'est une irréversibilité logique qui se traduit par une irréversibilité physique : la machine va dégrader de l'énergie, il faudra transformer de l'énergie électrique en chaleur [4] [5].

En général, un calcul est un graphe qui comporte des convergences. Dans le modèle de Turing [2] [3], un calcul qui s'arrête est un graphe qui ne comporte pas de divergence ni de boucle mais qui comporte des convergences [11] (figure 7).

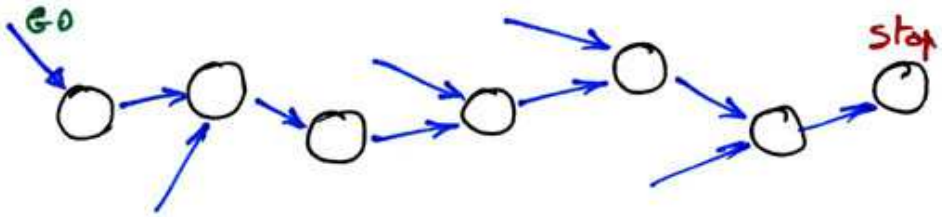

Fig. 7 – Un calcul qui s'arrête (sur une machine de Turing)

Pour prédire l'avenir d'une machine de Turing, il suffit de la laisser évoluer spontanément vers les cycles d'horloge ultérieurs. En revanche, pour prédire son passé, c'est en général

impossible (s'il y a des convergences dans son graphe). C'est conforme au second principe de la thermodynamique : c'est plus difficile de prévoir le passé que l'avenir ! C'est très différent des conceptions communes !

Nous venons d'évoquer une relation entre les convergences dans les graphes d'automates et la dissipation des ordinateurs. Cette façon de voir les choses est héritée de Landauer (1961) [4] et Bennett (1973) [5]. Elle semble dire que la dissipation est liée à l'utilisation de mémoires. Mais quel aspect des mémoires est concerné par la dissipation ? Bennett a surtout cherché à montrer qu'il s'agit de la perte d'information par effacement des mémoires. Il nous semble que ce serait plutôt l'aspect *synchronisation* effectuée par les mémoires : le fait qu'elles servent à créer un temps global et discret alors que le temps physique n'a pas la même structure [29].

Essayons un rapide historique de ces questions.

Si nous regardons le bilan énergétique d'un ordinateur, nous voyons que toute l'énergie électrique consommée est transformée en chaleur. De ce point de vue, un ordinateur est équivalent à un radiateur électrique (figure 8).

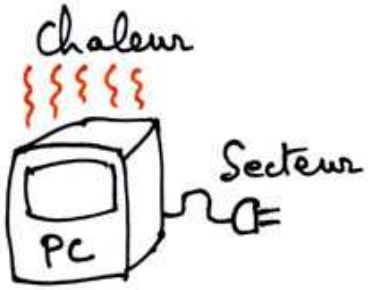

Fig. 8 – Bilan énergétique d'un ordinateur

Pourtant, la fonction d'un ordinateur est purement mathématique, il n'y a pas *a priori* d'enjeu énergétique. Le second principe de la thermodynamique ne nous dit rien sur un tel objet, sinon que la dissipation doit être positive ou nulle. Ne pourrait-on pas maîtriser, voire annuler cette dissipation ?

Cette question est liée à des enjeux industriels et économiques considérables. En effet, tous les appareils mobiles souffrent d'une autonomie restreinte, du poids des batteries, et tous les appareils électroniques souffrent du problème d'évacuation de la chaleur, et du coût de l'énergie. Dans la pratique, l'industrie a gagné un facteur 250 en 15 ans, à fonctionnalité égale, mais uniquement grâce à la miniaturisation des transistors.

La façon reconnue de traîter ces questions est de lier la dissipation aux convergences dans les graphes. Le lien est fait grâce au démon de Maxwell, objet qui est vu aujourd'hui comme une machine à calculer : automate qui cycle sur échantillonnage-effacement. L'effacement impose une convergence dans un graphe [6] [7] [8] [9] [10].

Bennett a montré en 1973 que dans tout calcul, on pouvait supprimer les convergences du graphe, c'est-à-dire rendre le calcul réversible [5]. Le graphe obtenu est linéaire (figure 9), c'est-à-dire sans divergence ni convergence.

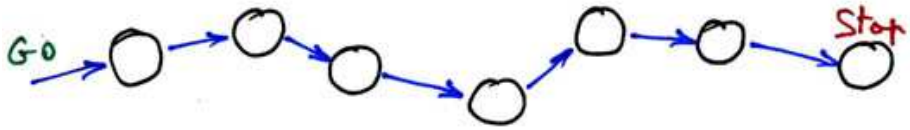

Fig. 9 – Graphe d'un calcul réversible

Du coup, le fait que ce soit celui d'un automate en temps discret n'est peut-être plus un point important car il n'y a plus de synchronisation à assurer. On peut se demander si cette absence de contrainte de synchronisation n'est pas le point essentiel [11].

Comment réaliser cette machine de Bennett ? Pour l'instant, personne n'y est parvenu (en 33 ans, malgré les enjeux considérables, ce qui prouve qu'il y a quelquechose qui résiste !) Une des voies est celle du calcul quantique [12], mais c'est bien difficile, il y a beaucoup de développements à faire, et cela demande de préciser beaucoup de choses en physique.

Fredkin et Toffoli en 1982 avaient proposé un modèle de calcul conservatif, celui du *Billiard Ball Computer* [13]. Il s'agit d'un modèle dans lequel toutes les fonctions logiques sont simulées par des mouvements et interactions de billes, interactions qui sont des chocs élastiques entre les billes et contre des miroirs (figure 10).

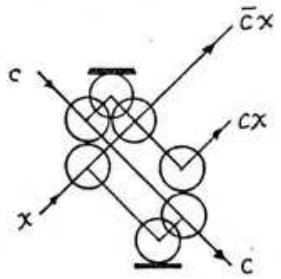

Fig. 10 – Ordinateur à boules de billard

On peut montrer que tous les calculs combinatoires peuvent être réalisés avec un tel procédé, de façon réversible et conservative.

Pourquoi ne pourrait-on pas le transposer en électronique ?

Une difficulté avec ce modèle est que le temps et l'espace y sont discrets, assurant ainsi une synchronisation implicite. La vitesse de chaque bille est la vitesse unité : une unité d'espace en une unité de temps. Or, on sait bien que l'espace et le temps de la physique ne sont pas discrets, depuis 2500 ans et les arguments de Zénon d'Élée [14] [15] (figure 11).

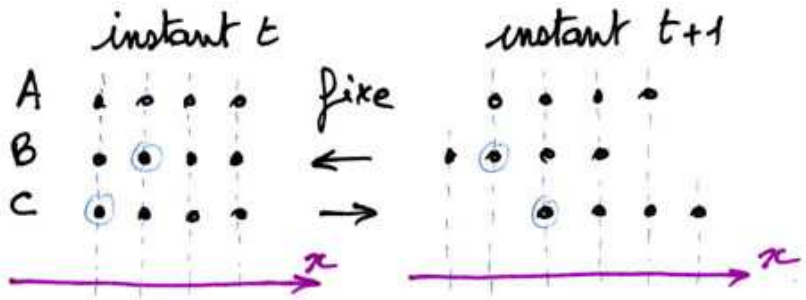

FIG. 11 – Quatrième argument de Zénon

Entre l'instant t et l'instant $t+1$, les rangées B et C se sont déplacées d'un cran, l'une vers la gauche, l'autre vers la droite. À quel instant se sont croisés les points entourés ? Il est nécessaire qu'il y ait un instant intermédiaire, sinon le mouvement serait impossible.

Aujourd'hui, nous dirions qu'il faut que le nombre qui représente le temps présente un *ordre dense* : entre deux nombres, il y en a un troisième. Ceci conduit à dire que dans une durée finie, il y a une infinité d'instants. (Ce n'était pas la problématique de Zénon qui, ne connaissant pas l'infini des mathématiques du 19ème siècle, s'en servait pour nier la multiplicité des étants, ainsi que la possibilité du mouvement, à la suite de Parménide [14].)

Lorsqu'on implémente l'ordinateur à boules de billard dans un espace-temps continu, les mouvements des billes ne s'effectuent pas à la vitesse exacte requise, les chocs n'ont pas les propriétés de symétrie souhaitées, et les trajectoires s'écartent des trajectoires idéales. Il faut prévoir un système de recalage des trajectoires qui demande des calculs supplémentaires et de la dissipation. Certains ont émis des raisonnements statistiques pour montrer que le recalage de la vitesse pourrait se faire d'une façon qui ne soit pas trop dissipative.

En électronique, une façon simple de synchroniser les bits qui circulent serait de faire un montage synchrone en intercalant, sur tous les chemins de données, des *bascules D* activées par une horloge commune. Mais cela serait dissipatif car introduirait un automate avec des convergences dans son graphe. Ce serait donc une synchronisation trop forte, qui aurait un effet contraire à l'effet recherché qui est d'annuler la dissipation.

Il semble que la question tourne autour de : "*Pourquoi et dans quelles circonstances avons nous besoin de synchroniser et que signifie synchroniser ? En quoi ce besoin de synchronisation serait-il lié à quelquechose d'intrinsèque à la nature du calcul effectué ?*"

2.3 Simulation d'un calcul par la physique de Newton

Essayons de comparer ce modèle de calcul (à boules de billard) avec la mécanique de Newton [16] (figure 12).

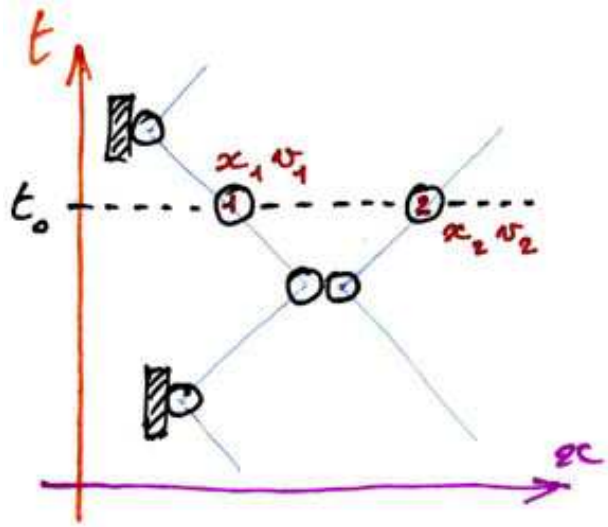

Fig. 12 – L'état chez Newton

Ce qui serait comparable à l'*état du calcul*, ce serait l'*état du système physique* à l'instant t_0, qui est entièrement défini par la connaissance de (x_1, v_1, x_2, v_2), positions et vitesses des billes.

Dans un automate, le *mouvement*, c'est-à-dire l'évolution temporelle, est régi par la fonction de transition (c'est-à-dire le graphe) qui indique comment on passe de l'état présent à l'état de l'*instant suivant*. Mais dans un ordre dense, *il n'y a pas d'instant suivant* ! La notion de *successeur* n'est pas compatible avec un ordre dense. Alors, le calcul dans l'espace-temps de Newton prend la forme d'une équation intégrale : *le calcul dans l'espace-temps de Newton est le calcul intégral.*

On peut relier ça à un autre argument de Zénon, le premier (figure 13). Pour franchir une distance finie (ici de A à B), il faut déjà parcourir la moitié de cette distance, puis la moitié de ce qui reste, etc., c'est-à-dire qu'il faut une infinité d'instants.

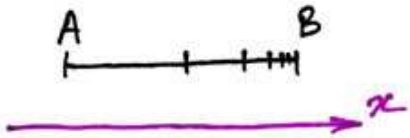

Fig. 13 – Premier argument de Zénon

Zénon en concluait qu'il faudrait un temps infini. Mais nous savons, nous, que :

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + ... = 1$$

Nous voyons bien là que si nous savons calculer une intégrale, nous obtenons un temps

fini compatible avec la mécanique de Newton. Si en revanche, nous considérons que chaque addition d'un terme de la série doit prendre une étape de calcul en temps discret, alors nous retombons sur la contradiction pointée par Zénon.

Dans un ordinateur synchrone, ce sont les bascules et l'horloge qui nous servent à simuler le temps discret de l'automate à l'aide du temps de la physique. Essayons d'illustrer cela (figure 14). CK est le signal d'horloge, B_1, B_2 et B_3 sont les *bascules D* alors que Op_1, Op_2 et Op_3 sont les opérateurs logiques.

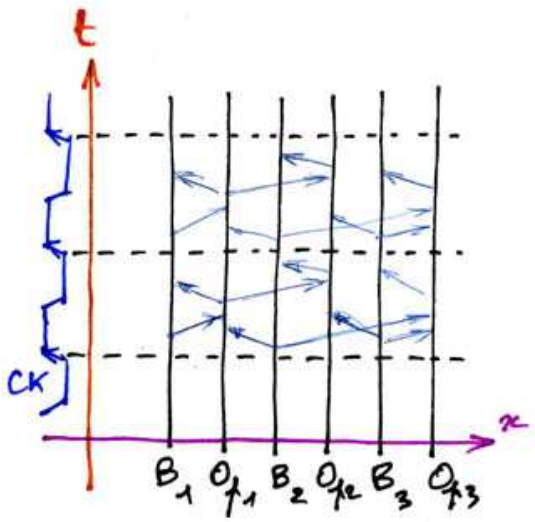

Fig. 14 – Ordinateur newtonien

Le signal d'horloge CK crée un temps discret. Mais il fait autre chose : en répandant l'horloge dans tout l'espace, il crée un temps global, *ie* qui est le même partout, ce qui est compatible avec cet aspect du temps chez Newton.

Cette structure de temps, à la fois discret et global, permet l'implémentation physique des automates car elle permet de créer une bijection entre les états logiques de l'automate et les états mécaniques du système physique. En outre, cette bijection permet de relier la notion d'*information*, fondée sur des statistiques de transitions d'automate [17], avec la notion d'*entropie* en physique. Il faut bien voir que les définitions de ces deux notions reposent sur la notion d'*état*, classe d'équivalence de passés, étroitement reliée à un modèle de temps global.

3 Circuits asynchrones et physique relativiste

3.1 Limitation du modèle newtonien

Notre ordinateur newtonien, avec sa synchronisation par une horloge globale périodique, ressemble de très près à un orchestre synchronisé par la baguette de son chef. À ceci près que les ordres de grandeur sont très différents :

- Signal sonore / signal lumineux (rapport de vitesses : 1 million)
- Mesure de deux secondes / période de une nanoseconde (rapport de durées : 2 milliards)
- Synchronisation par la lumière / synchronisation par la lumière (rapport : 1)

La troisième comparaison est troublante. En effet, synchroniser des signaux sonores à l'aide de signaux lumineux, cela fonctionne bien. Mais synchroniser des signaux lumineux à l'aide de signaux lumineux peut poser des problèmes. De fait, ce qui fonctionnait bien pour les ordinateurs lorsque les fréquences d'horloge étaient plus faibles, a du mal à fonctionner aujourd'hui où les fréquences d'horloge dépassent le gigahertz. Reprenons le schéma précédent mais en considérant cette fois que le signal CK de synchronisation est un signal comme les autres (figure 15).

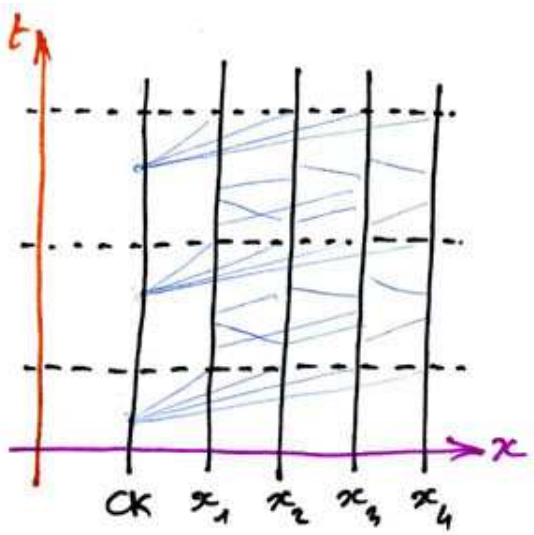

Fig. 15 – Explicitons la synchronisation

Nous voyons sur cette figure que les instants de synchronisation, c'est-à-dire les lignes horizontales en pointillés, sont tout-à-fait imaginaires. Elles ne correspondent pas à un "*en même temps*" qui serait de nature physique. Elles indiquent simplement un ordre total entre les événements qui sont tous rangés suivant un indice entier qui est le numero de la période d'horloge. Localement, il importe uniquement que le signal d'horloge arrive après le signal qu'il échantillonne. L'horloge globale, avec sa période choisie suffisamment grande, installe cet ordre total.

Dès que nous prenons en compte la finitude de la vitesse de la lumière, dire que le temps est le même partout n'a aucun sens. La preuve, c'est que pour obtenir une relation d'ordre total, il faut le fabriquer explicitement (avec l'horloge) car ce n'est pas la nature qui nous l'offre ! Or, prendre en compte la finitude de la vitesse de la lumière, et considérer qu'il n'y a pas de temps global, c'est se placer dans un espace-temps relativiste.

La vitesse de la lumière dans le vide est très faible : 30 centimètres par nanoseconde. À la surface d'une puce de silicium, c'est encore pire, de l'ordre de 1 cm en plusieurs nanosecondes. Avec les technologies actuelles, le schéma est plutôt celui de la figure 16. Le temps de propagation du signal d'horloge est de l'ordre de plusieurs périodes.

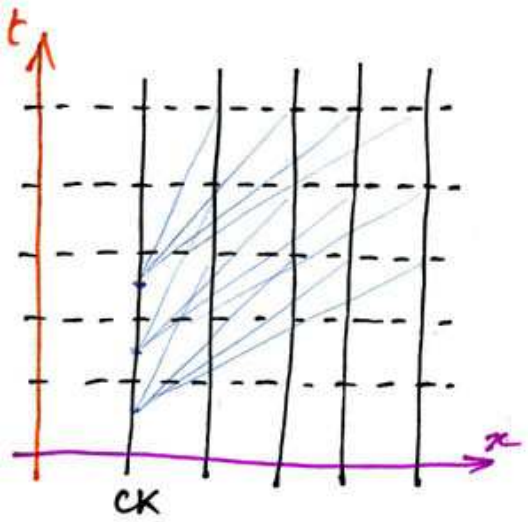

Fig. 16 – Lumière lente

Dans les puces synchrones modernes, ça devient très complexe. L'horloge est envoyée à des centaines de millions de bascules, à travers un arbre equilibré. Elle est responsable de la majeure partie de la dissipation, du coût de Silicium, du coût de développement. Ce coût énorme provient de la volonté de fabriquer un espace-temps newtonien logique à l'aide d'un espace-temps physique qui n'est pas newtonien. Cela revient à dire que la relativité ne concerne pas que les mouvements de étoiles, mais pénètre très intimement notre vie quotidienne.

Pourquoi ne pas essayer de supprimer l'horloge ? Avons-nous vraiment besoin de cet ordre total et du temps discret ? Et si on se contentait d'un ordre partiel ? d'un ordre d'une autre nature ? (figure 17)

Entre les deux chemins de couleurs différentes, nous avons besoin d'un ordre local : un signal doit arriver avant l'autre (ici le signal à échantillonner doit arriver avant l'horloge). Si nous décidons de ne plus simuler un temps discret et global, il va falloir que le concepteur du circuit compare tous les chemins deux-à-deux, ce qui peut l'entraîner vers quelquechose de très complexe.

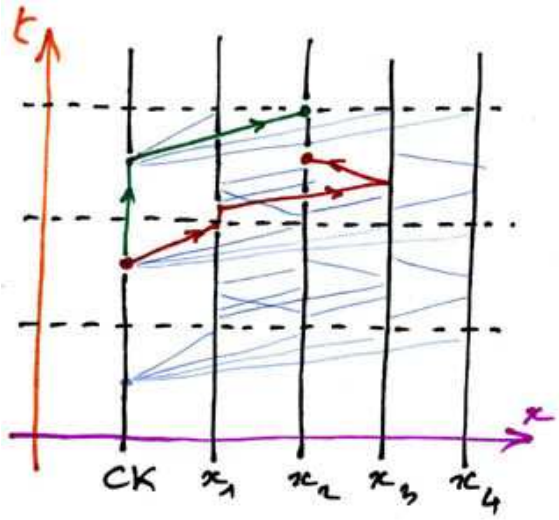

Fig. 17 – Ordre entre deux chemins dans l'espace-temps

Nous voyons apparaître là, *a contrario*, deux avantages de l'horloge :

- L'ordre total du temps discret global évite à l'ingénieur d'avoir à considérer toute la combinatoire des comparaisons de chemins : il suffit que chaque chemin de propagation soit plus court que la période d'horloge.
- Les propagations sont de nature physique et non logique. Quand la condition précédente est réalisée, le concepteur peut se contenter de ne considérer que les aspects logiques du calcul (dans une première phase de conception qui ne regarde pas la performance). Nous séparons ainsi les aspects logiques des aspects physiques.

Ces avantages sont bien sûr à balancer avec les inconvénients cités auparavant. Mais nous pouvons nous poser la question suivante :

Serait-il possible de trouver un modèle de calcul distribué, qui corresponde à un ordre partiel, et qui ne demanderait pas de considérer toute la combinatoire des chemins, en permettant ainsi de garder une bonne séparation entre la conception physique et la conception logique des circuits ?

Il semble que la réponse soit positive, avec les circuits *asynchrones DI* (Delay-insensitive) [18] [19] [20], que nous allons décrire ci-après. On réalise déjà des circuits avec cette méthode, mais une bonne formalisation, qui simplifierait l'automatisation de la conception, n'est pas encore clairement établie.

3.2 Quelques mots sur la relativité

Pour dire des choses très simples sur la relativité, considérons deux personnes qui se regardent (figure 18).

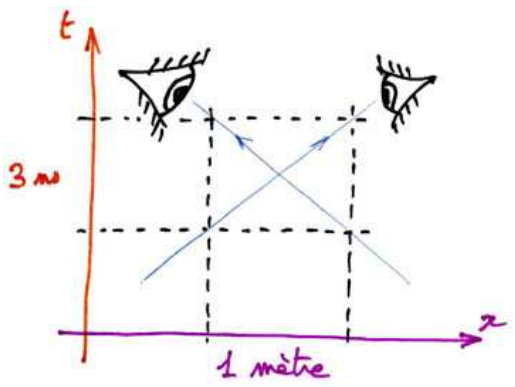

FIG. 18 – Deux personnes se regardent

Il est clair que :
– La simultanéïté ne peut être définie que localement.
– Une "*coupe de l'espace-temps à l'instant t*" ne veut rien dire (pas de temps global).
– L'ordre total temporel n'existe pas.
– Le temps n'est pas un nombre.
– On peut avoir une horloge locale (temps nombre réel local) pour chaque lieu.
Il n'y a pas besoin de considérer des objets en mouvement rapide pour avoir ces propriétés. Elles découlent du fait que tout se propage à vitesse finie, même la lumière.

Pour raisonner sur une coordonnée temporelle qui ne soit pas uniquement locale, Einstein a proposé un *algorithme* de synchronisation (conventionnel !) (figure 19) [21] [22].

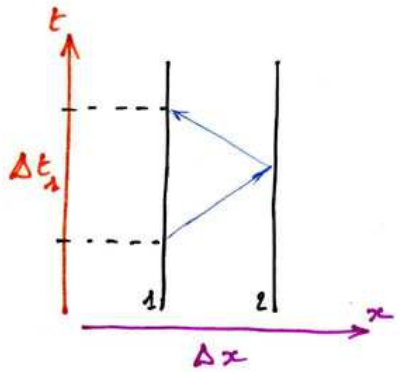

FIG. 19 – Synchronisation d'Einstein

On considère que chaque objet emporte une horloge. Si ces horloges dialoguent, elles peuvent se synchroniser après avoir calculé la distance qui les sépare (la constante c est fixée par convention) :

$$\Delta x_1 = c\frac{\Delta t_1}{2}$$

Le couple $(\Delta x, \Delta t)$ est local à un objet (sa notation est indexée par le numero de l'objet qui fait le calcul) et diffère d'un objet à l'autre si les objets sont en mouvement, car la valeur de c est choisie la même en toutes circonstances, par convention. Si on regarde comment passer d'un couple à l'autre, on obtiend les formules de Lorentz, formules de changement de référentiel.

C'est cette façon de relier les temps locaux qui s'appelle *synchronisation*. C'est la création, par un algorithme, d'une cohérence entre les temps locaux. *C'est l'algorithme qui engendre les référentiels*, et donc la formulation des lois physiques qui va en découler.

Il faut bien avoir présent à l'esprit ce qu'il y a de conventionnel et de "*fabriqué par l'homme*" dans la structure de cet espace-temps. La construction de cet espace-temps est algorithmique. Ce n'est que parce que cet algorithme est compatible avec les relations de Lorentz (relations de changement de repère entre repères en mouvement), et que ces relations de Lorentz sont connues comme compatibles avec les expériences, que cet algorithme a pu être considéré comme construisant un espace-temps compatible avec celui de la nature, supposé objectif par principe.

Ceci est important pour comprendre comment fonctionne actuellement la *fabrication* de la *coordonnée temps* du TAI (temps atomique international), ou pour comprendre ce que fait le GPS (Global positioning system). Il faut bien voir que ce qu'on appelle *mesure* est en fait ce qui est *produit* par un algorithme distribué [23] [24] [25].

Qu'est-ce que se positionner dans l'espace-temps à 4 dimensions (par exemple longitude, lattitude, altitude, date) ?

Lorsque nous utilisons le système GPS, le récepteur écoute au moins 4 satellites, qui délivrent des repères temporels. Le récepteur calcule quelle position peut bien être compatible avec les écarts temporels des signaux reçus. Les repères temporels des satellites sont eux-mêmes produit par un calcul distribué en interaction avec des horloges atomiques, qui cherche à établir une cohérence entre les indications de ces horloges. Cette cohérence est obtenue en tenant compte de toute la physique connue sur l'espace-temps (relativité générale). L'ensemble est un algorithme distribué dont le produit est une synchronisation.

En d'autres termes, *mesurer* une position dans l'espace-temps, c'est exécuter un algorithme de *synchronisation*, c'est *fabriquer* l'espace-temps.

En ce qui nous concerne, nous ne retiendrons que ce qui a rapport à la causalité et l'ordre partiel qui en résulte. Considérons, figure 20, la propagation d'un événement de A à B, par exemple un front sur un signal électrique.

Nous pouvons utiliser cette propagation d'événement pour définir une relation d'ordre causal, car cet ordre est conservé par changement d'observateur, même en mouvement (intervalle de genre temps dont le signe est conservé par les relations de Lorentz).

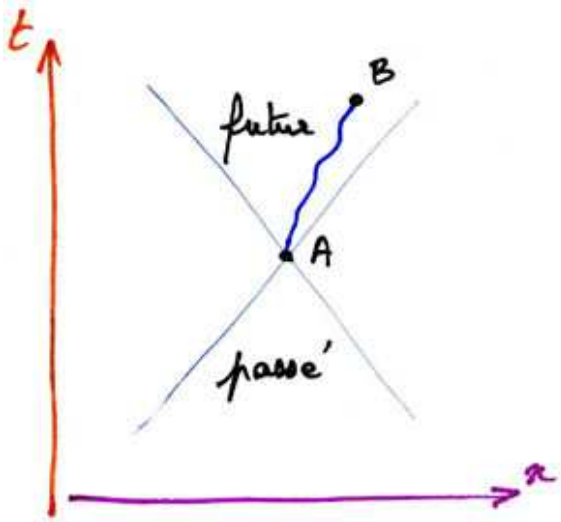

Fig. 20 – Causalité dans un cône de lumière

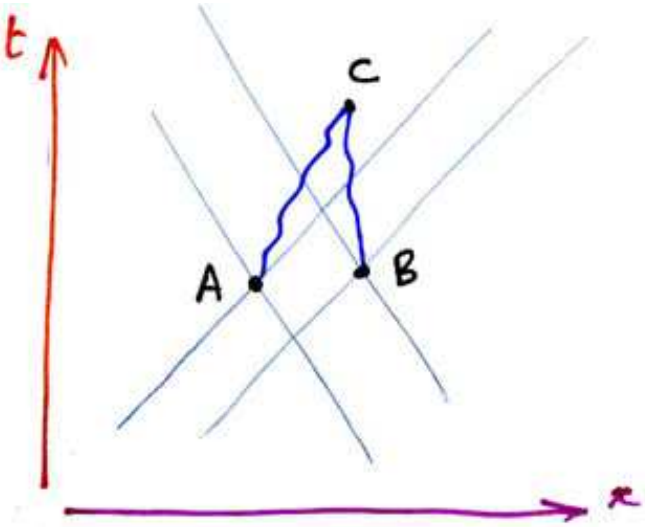

Fig. 21 – Ordre partiel et causalité

Considérons, figure 21, deux propagations qui atteignent un même point de l'espace-temps. C est postérieur à A et postérieur à B, mais il n'y a pas de relation d'ordre entre A et B.

Cette relation d'ordre partiel, indépendante de l'observateur puisque conservée dans les mouvements de référentiels, peut être utilisée pour implémenter un modèle de calcul distribué. C'est ce que nous allons voir maintenant avec le modèle des circuits asynchrones [26] [28].

3.3 Circuits asynchrones

Nous allons juste donner ici quelques exemples concernant les circuits *asynchrones DI* (*Delay-Insensitive* : Insensibles aux retards de propagation). La règle du jeu est de trouver des composants et des montages dont la fonction logique globale, liée à un ordre causal partiel, est indépendante des temps de propagation.

Considérons, afin de nous en écarter ensuite, la transmission d'une donnée sur un bit en synchrone (figure 22).

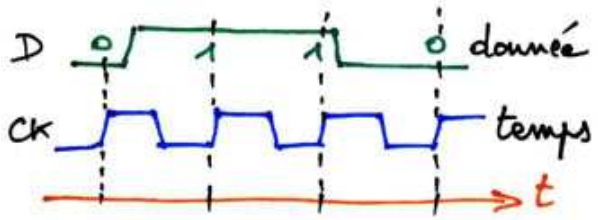

Fig. 22 – Réception synchrone d'une donnée

On ne peut qu'être frappé ici par la dissymétrie de rôle entre les deux signaux. Dissymétrie, non de nature physique, mais dans l'énoncé logique de leur rôle. Pourquoi un des deux signaux (CK) serait-il plus *temporel* que l'autre ? Pourquoi l'un des deux (D) serait-il plus une *information* que l'autre ?

Quoiqu'il en soit, nous souhaitons supprimer l'horloge. Mais comment, sans horloge, faire la différence entre les suites de bits suivantes ? (figure 23).

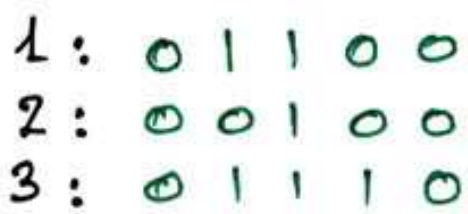

Fig. 23 – Trois séries de données

Nous allons considérer que toute opération logique est un dialogue entre un composant et son

environnement, et que tout message doit obtenir une réponse (doit être acquitté). Prenons l'exemple d'un XOR (OU exclusif) (figure 24).

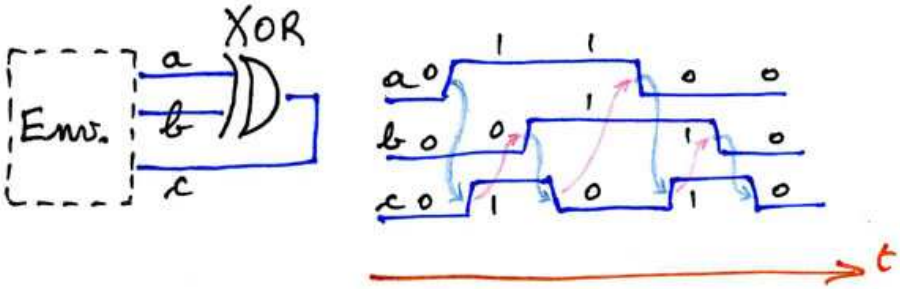

FIG. 24 – XOR et son environnement

On peut regarder ce schéma de plusieurs façons. Si on regarde les niveaux logiques, on retrouve la fonction booléenne habituelle du $OUexclusif$.

Mais nous avons également dessiné les liens de causalité (flèches courbes), en bleu ceux qui sont de la responsabilité du composant XOR, en rouge ceux qui sont de la responsabilité de l'environnement. Nous pouvons maintenant constater que les flèches bleues et rouges alternent. Cela correspond au dialogue et à l'acquittement que nous avons mentionné. Nous pouvons en conclure tout de suite une contrainte sur l'environnement : ce dernier ne doit pas modifier plus que une de ses sorties (a ou b) à la fois, sinon l'acquittement par c est impossible car le XOR ne pourrait pas savoir si son environnement veut modifier une ou les deux de ses sorties. Nous voyons donc que le caractère *exclusif* de cet XOR est de la responsabilité de son environnement (et non de la responsabilité de l'objet XOR) !

Nous venons en fait de déplacer la notion de *fonction du composant* XOR vers une notion de *fonction du couple composant/environnement*, plus précisément même vers la notion de *fonction de l'interface* entre les deux.

Nous pouvons reprendre notre dessin en ne conservant que les flèches de causalité, et en le déformant pour le transformer en diagramme d'espace-temps (figure 25).

Les flèches bleues et rouges sont maintenant situées sur la trajectoire verticale de chaque composant et représentent leur action propre. Les flèches obliques dessinées représentent la propagation des signaux de communication. La relation d'ordre partiel concerne ces deux types de liens causaux : le long de la trajectoire verticale des composants (évolution temporelle locale de chaque composant) et le long d'une oblique (propagation d'un événement qui est un front électrique). La fonction du XOR est maintenant représentable par la *trace* suivante :

$$*[(a|b); c]$$

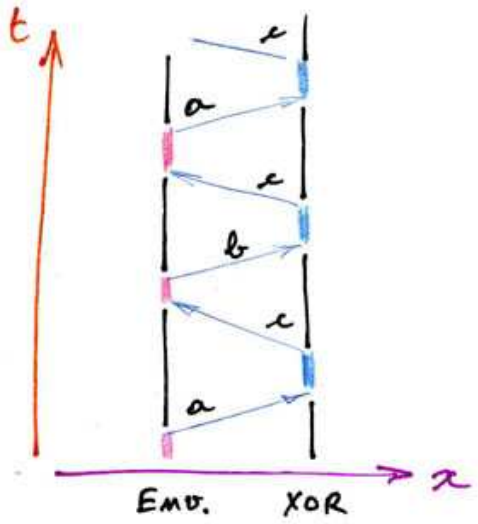

Fig. 25 – Interactions du XOR

où
– | représente *un et un seul* (entre les deux symboles à droite et à gauche)
– ; représente la *concaténation* (succession temporelle et causale, ordonnée)
– $*$ représente la *répétition* (concaténation répétée du même motif).
Passons maintenant à un autre opérateur : le *Rendez-vous* (figure 26).

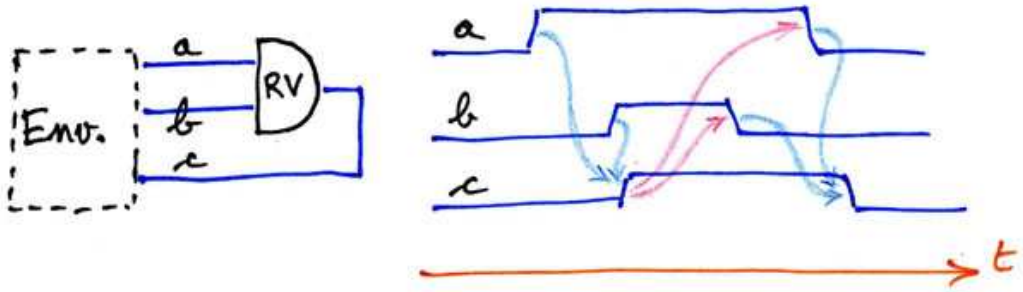

Fig. 26 – Rendez-vous

La sortie c ne change que si les deux entrées ont changé, chacune une fois et une seule. Le diagramme d'espace-temps est le suivant (figure 27). (Nous avons séparé l'environnement en deux parties.)

Chaque environnement ne change qu'une fois et une seule après un changement de c. La trace est :

$$*[(a||b); c]$$

où le symbole || indique : *présence des deux événements à droite et à gauche, chacun une fois et une seule*. Remarquons ici que l'ordre des événements de la figure 27 est un ordre partiel

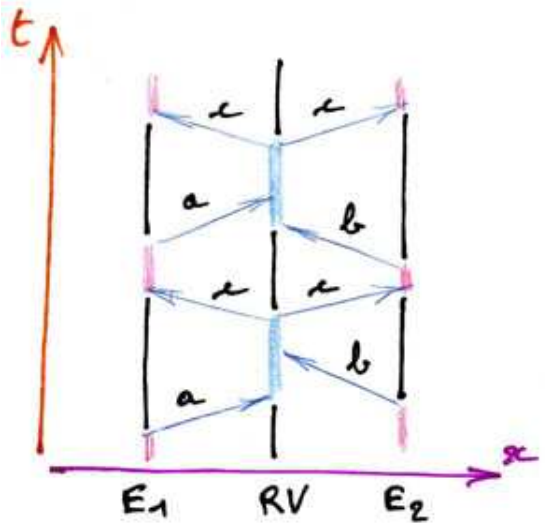

Fig. 27 – Diagramme du Rendez-vous

et non total, c'est-à-dire que cet ordre peut être vu différemment par deux observateurs. En revanche, l'écriture de la trace est compatible avec la vision de tous les observateurs.

Pouvons nous faire le même genre de description pour le *ET* logique ? (figure 28).

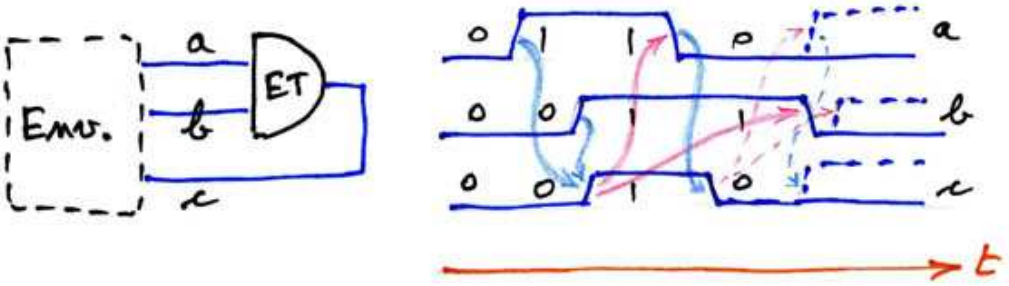

Fig. 28 – *ET* logique

Nous voyons qu'il y a un croisement entre une flèche rouge et une flèche bleue, ce qui correspond à un acquittement incomplet, qui va entraîner ensuite une concurrence entre deux flèches rouges, et qui fait que la suite du calcul sera dépendant des temps de propagation. Cet opérateur n'est donc pas utilisable dans un contexte *DI*. Il en est de même pour le *OU* logique.

Nous pouvons faire deux remarques :
- Les opérateurs logiques *DI* ne sont pas les mêmes que ceux de la logique synchrone.
- Les opérateurs ne sont plus décrits en terme d'opérations booléennes, mais en terme de liens de causalité entre des événements, qui sont ici les fronts électriques, mais qui pourraient être de nature physique toute autre.

Passons tout de suite à un assemblage de composants : le diviseur par trois de Ebergen

(figure 29).

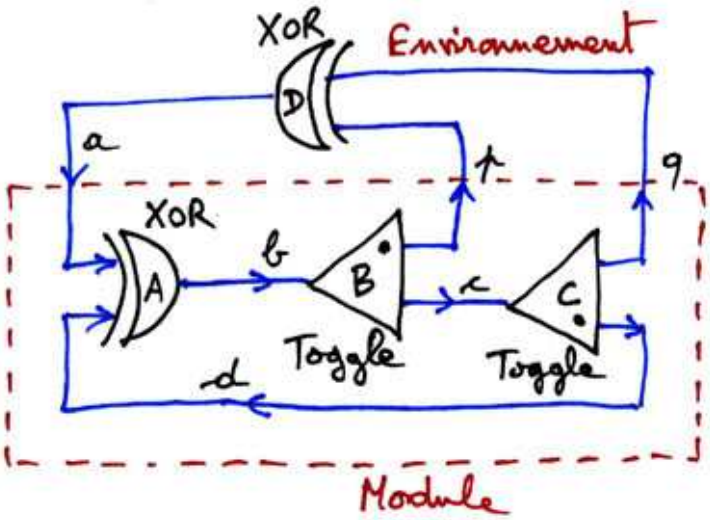

Fig. 29 – Diviseur par trois

Considérons la trace complète de tous les événements (la partie qui correspond à une période puisque c'est périodique) ainsi que la projection de cette trace sur chacun des composants. Commençons arbitrairement par a (figure 30).

trace complète	:	a	b	f	a	b	c	d	b	f	a	b	c	g
trace de A	:	a	b	.	a	b	.	d	b	.	a	b	.	.
trace de B	:	.	b	f	.	b	c	.	b	f	.	b	c	.
trace de C	:	.	.	.	.	.	c	d	.	.	.	.	c	g
trace sur ----	:	a	.	f	a	.	.	.	.	f	a	.	.	g

Fig. 30 – Trace complète et traces projetées

Les traces projetées sont obtenues simplement en supprimant de la trace complète les symboles qui ne concernent pas le composant particulier qu'on regarde.

C'est la trace sur la frontière en pointillés qui donne le nom à ce montage : *diviseur par trois*. Le diagramme espace-temps est le suivant (figure 31).

Bien que cet exemple soit trop simple pour mettre en évidence toutes les propriétés de ces assemblages (en particulier cet exemple n'exhibe pas de paraléllisme), on peut montrer que ce genre de montages permet de faire tous les calculs, à l'aide d'un jeu de composants primitifs très réduit. Ces composants primitifs trouvent des réalisations à base de quelques transistors. Il existe des circuits commercialisés qui utilisent ce type de calcul.

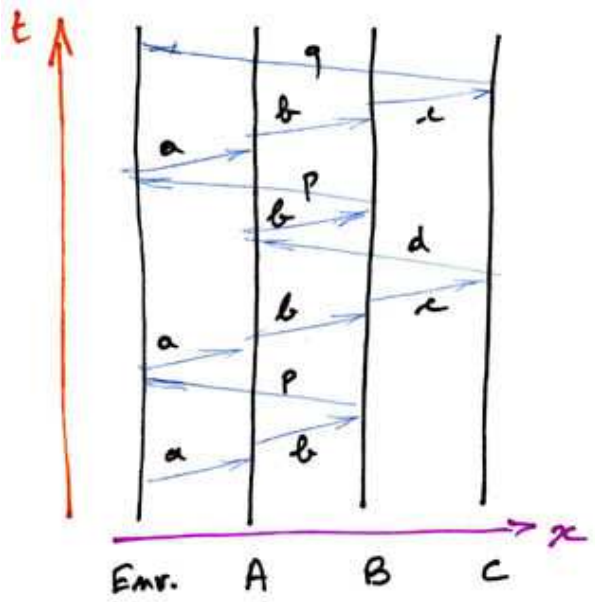

FIG. 31 – Diagramme espace-temps du diviseur par 3

Ces circuits ont des propriétés très séduisantes :
- Ça se synchronise "tout seul", sans horloge globale, en réalisant l'ordre partiel décrit par la fonction logique.
- Sur une puce de Silicium, la fonction ne dépend pas du placement des composants, seule la performance en dépend. On peut ainsi séparer d'une part la conception logique de la fonction et d'autre part son optimisation. C'est une séparation entre les aspects logiques et les aspects physiques, semblable à celle que procurait l'horloge globale pour les circuits synchrones. On peut en outre faire une conception modulaire ou incrémentale, ce qui n'était pas possible en logique synchrone, où toutes les parties devaient respecter la même période d'horloge.
- Du point de vue de la dissipation, la modélisation est simple, il suffit de compter le nombre d'événements impliqués dans le calcul, c'est-à-dire le nombre de symboles qui se sont propagés [27] [29]. En général, la dissipation est bien moindre qu'en synchrone car il n'y a pas de transition inutile, et il n'y a pas la dissipation due à l'horloge.

Par ailleurs, les traces qui décrivent les fonctions se traduisent de façon simple en diagrammes d'espace-temps relativistes, ce qui permet de voir que ces fonctions de calcul distribué sont compatibles avec la géométrie de notre espace-temps physique. Une des conséquences est que le calcul est insensible à des déformations spatiales du réseau, en particulier les composants peuvent être *mobiles*, et à des vitesses très élevées, puisque la relation d'ordre causale est compatible avec celle de la mécanique relativiste.

À côté de ces belles propriétés, il subsiste de nombreuses questions :
- Ces circuits asynchrones *DI* définissent un modèle de calcul distribué. Mais sa formalisation n'est pas encore achevée. Il serait souhaitable d'avancer dans cette direction afin de mieux comprendre de quelle notion de calcul il s'agit.
- Comment comparer ce modèle de calcul distribué à d'autres, mais aussi au modèle de

Turing ? A-t-il un pouvoir d'expression supérieur ? Si oui, comment simuler les circuits en cours de conception ? Sur quelle machine ?
- Il n'y a plus de notion d'*état* de la machine. Comme il n'y a plus d'ordre total, la notion même de configuration à un instant donné semble disparaître, car elle dépend de l'observateur.
- La notion d'*information* ne peut plus être celle de Shannon, car celle-ci est trop liée aux automates et à leur notion d'état global. Les notions d'entropie et de dissipation, trop liées à une thermodynamique newtonienne ne conviennent plus.
- La notion de *signal* est aussi bousculée. Ce qui était une valeur locale fonction du temps doit probablement être remplacé par une notion d'*événement dans l'espace-temps*, qui serait un segment de ligne de propagation, pris dans une boucle algorithmique ?

En outre, la notion d'*objet* change également. Regardons le schéma du *diviseur par trois*. Quelle partie *réalise* (au sens de *est reponsable de*) la fonctionalité ?
- Ce qui est à l'intérieur des pointillés ? Pas tout seul, car si l'environnement envoie a alors qu'il n'a pas reçu p ou q, il va y avoir des interférences de calcul !
- L'extérieur ? Mais, à l'extérieur, il n'y a qu'un XOR !
- La collaboration des deux ? Certes, mais il y a plein d'environnements possibles, compatibles avec ce module et cette fonction !

On ne peut pas voir ces objets-là comme un jeu de *Lego* dans lequel les composants sont les briques de base d'un assemblage, chaque brique ayant une fonction, indépendante de son environnement. Les fonctions sont définies aux interfaces, et s'expriment sous forme de traces. D'un côté d'une interface, on peut faire des substitutions compatibles avec la trace de l'interface : soit des simplifications, soit au contraire en ajoutant des composants pour obtenir un calcul plus complexe.

4 Conclusion

Nous avons principalement exploré les différences entre les structures d'ordinateurs synchrones (séquentiels) d'une part et distribués (asynchrone DI) d'autre part. Nous avons vu que cette distinction est comparable à la distinction en physique entre espace-temps newtonien et espace-temps relativiste.

Dans le cas des ordinateurs synchrones, c'est l'horloge (et les mémoires) qui sert à créer un temps global alors que le temps de la physique n'a pas cette propriété. Il apparaît clairement ainsi que cette méthode de *synchronisation* est précisément ce qui permet de *simuler* une structure de temps à l'aide d'une autre.

Ceci nous conduit à généraliser l'usage du mot *simulation* à toute implémentation d'ordinateur : plutôt que de considérer que le monde logique et le monde physique ne sont pas de nature comparable, nous préférons considérer des deux côtés leur structure logique (ils sont tous les deux décrits en un langage), et envisager la *réalisation* comme une *simulation*. Du côté physique, le langage mathématique est celui des lois de la nature, ou bien, à un niveau

plus macroscopique, celui des caractéristiques des transistors pour un électronicien.

D'ailleurs, les électroniciens ont coutume de simuler, à l'aide d'un logiciel sur un ordinateur classique, le fonctionnement d'un futur circuit en cours de conception. Ceci est possible si la puissance d'expression de la machine à réaliser n'est pas supérieure à Turing, ce qui est assurément le cas pour toutes les machines séquentielles synchrones. Dans ce cas, un modèle simplifié de la physique est intégré au simulateur logiciel. Il doit prendre en compte une notion de simultanéïté (lors des transferts simultanés entre mémoires, par exemple dans le cas d'un registre à décalage), qui est simple à réaliser pour le cas séquentiel car on se ramène alors aisément à un temps global.

Mais lorsqu'on envisage de réaliser des circuits asynchrones, dans lequel l'ordre partiel ne permet pas de disposer d'une notion de simultanéïté, ce qui laisse envisager que le pouvoir d'expression pourrait éventuellement être supérieur à Turing, comment peut-on envisager une simulation logicielle sur un ordinateur séquentiel ?

Il nous semble que le point difficile de toute simulation concerne le changement de modèle de temps. Les techniques utilisées s'appellent alors *synchronisation*. Dans ce cadre, les mots *réalisation*, *simulation*, *synchronisation* prennent un sens très voisin. Il nous semble alors que les trois disciplines :
– comparaisons des modèles logiques de calcul distribué,
– implémentation des ordinateurs,
– et exploration des lois de la physique
sont bien proches ! Elles consistent toutes les trois à définir des langages logiques de machines distribuées et à chercher à simuler chacune de ces machines par les autres.

Références

[1] Shannon, C. E., *A Symbolic Analysis of Relay and Switching Circuits*, AIEE Transactions, 57, 713–723, (1938).

[2] Turing A. M., *On Computable Numbers, with an Application to the Entscheidungs Problem*, Proceedings of the London Mathematical Society Ser 2, Vol. 42, p. 230, (1936) et Vol. 43, p. 544-546, (1937).

[3] Minsky M. L., *Computation : Finite and Infinite Machines*, Prentice-Hall, (1967).

[4] Landauer R., *Irreversibility and Heat Generation in the Computing Process*, IBM J. Res. Develop. (1961) 183-191.

[5] Bennett C. H., *Logical Reversibility of Computation*, IBM J. Res. Develop. (1973) 525-532.

[6] Maxwell J. C., *Theory of Heat*, Text-books of Science, (London : Longmans, Green Co) 4th Ed. (1875) 328-329 (1st Ed. 1871).

[7] Szilard L., *Über die Entropieverminderung in einem thermodynamischen System bei Eingriffen intelligenter Wesen*, Zeit. Phys. **53** (1929) 840-856 ; English translation : *On the*

Decrease of Entropy in a Thermodynamic System by the Intervention of Intelligent Beings, Behavorial Science **9** (1964) 301-310.

[8] Bennett C. H., *Demons, Engines and the Second Law*, Scientific American (1987) 88-96.

[9] Gabor D., *Light and Information, Richie lecture, University of Edimburg, 1951*, Progress in Optics **1** (1961) 109-153.

[10] Brillouin L., *La Science et la Théorie de l'Information*, (Masson et Cie, 1959) (Editions Jacques Gabay, 1988).

[11] Matherat P., Jaekel M.-T., *Dissipation logique des implémentations d'automates - dissipation du calcul*, TSI (Technique et Science Informatiques), Vol. 15, numero 8, (1996), p. 1079-1104. http://www.comelec.enst.fr/~matherat/publications/tsi96/ et en anglais à http://fr.arxiv.org/abs/quant-ph/9805018

[12] Deutsch D., *Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer*, Proc. R. Soc. **A400** (1985) 97-117.

[13] Fredkin E. and Toffoli T., *Conservative Logic*, International Journal of Theoretical Physics, 21(3/4) :219-253, (1982).

[14] Aristote, *Physique*, livre VI, chap. 9.

[15] Russell B., *Our Knowledge of the External World*, (1914), réédition Routledge 2000.

[16] Newton, I., *Principia*, University of California Press, 1962, First ed. Philosophiae Naturalis Principia Mathematica (1687).

[17] Shannon C. E. and Weaver W., *The Mathematical Theory of Communication*, Illinois University Press, (1949).

[18] Udding J. T., *A formal model for defining and classifying delay-insensitive circuits and systems*, Distributed Computing, vol 1, pages 197-204, (1986).

[19] Ebergen, J. C., *Translating Programs into Delay-Insensitive Circuits*, thesis, Technische Universiteit Eindhoven (October 1987).

[20] Ebergen Jo. C., *A formal approach to designing delay-insensitive circuits*, Distributed Computing, vol 5, pages 107-119, (1991).

[21] Einstein, A., *Zur Elektrodynamik bewegter Körper*, Annalen der Physik, 17, 891–921, (1905).

[22] Russell B., *ABC of Relativity*, (1925), 5ème réédition Routledge 1993.

[23] Lewandowski, W. and Thomas, C. (1991) GPS Time Transfer. In *Special Issue on Time and Frequency : Proceedings of the IEEE*, 991–1000.

[24] Guinot, B. (1997) Application of general relativity to metrology. *Metrologia* **34**, 261–290.

[25] Quinn, T. J. (1991) The BIPM and the Accurate Measurement of Time. In *Special Issue on Time and Frequency : Proceedings of the IEEE*, 894–905.

[26] Russell B., *The Analysis of Matter*, (1927), réédition Routledge 1997.

[27] Ebergen Jo. C., Segers J. and Benko I., *Parallel Program And Asynchronous Circuit Design*, in *Asynchronous Digital Circuit Design*, Workshop in Computing, G. Birtwistle and A. Davis editors, pages 50-103, BCS Springer,(1995).

[28] Matherat P. et Jaekel M.-T., *Concurrent computing machines and physical space-time*, Mathematical Structures in Computer Science , Vol. 13, numero 5, (oct. 2003), p. 771-798, CUP. http://www.comelec.enst.fr/~matherat/publications/mscs03/

[29] Matherat P., *Où en est-on de la dissipation du calcul ? Retour à Bennett*, https://hal.ccsd.cnrs.fr/ccsd-00082436, juin 2006.