Calculer avec des phases

Philippe Matherat

CALCULER AVEC DES PHASES

Philippe Matherat

Résumé

Nous présentons une élaboration qui se veut une tentative de définition d'un modèle de calcul compatible avec un temps qui ne serait pas un nombre mais un ordre partiel. Initialement, nous avons été motivés par le cadre des circuits "sans horloge". Nous attendons de cette théorie une compréhension des phénomènes de synchronisation dans les systèmes distribués, dans ce qu'ils ont de propre à chaque calcul particulier. Nous en attendons également une compréhension des phénomènes de dissipation logique qui seraient nécessairement liés à chaque calcul. Ce questionnement nous a conduit à des remises en cause profondes des notions habituelles d'espace, de temps, de composition d'objets, en liaison avec le calcul.

1 Objectifs, introduction

Ce texte est une étape dans une tentative pour mettre en place une théorie, un modèle de calcul, dans lequel le temps ne serait pas un nombre entier (comme il l'est dans les circuits synchrones ou les automates), ni même un nombre réel (qui pourrait permettre un ordre total entre les événements du calcul), mais un ordre partiel. Cette tentative est motivée par le cadre nécessaire aux circuits asynchrones (ou plutôt "clockless", c'est-à-dire "sans horloge"), qui se décrivent avec un tel modèle de temps. Par ailleurs, ce modèle de temps est compatible avec celui de la relativité, ce qui constitue un attrait supplémentaire puisque compatible avec une notion de temps de la physique.

Ce que nous attendons de cette théorie est une compréhension des phénomènes de synchronisation dans un système distribué. Par le mot "synchronisation" nous n'entendons pas "ce qui fait référence à une horloge commune", puisque notre modèle ne comporte pas de temps commun, global au système. Mais nous entendons "l'ordre partiel engendré par les communications entre les composants du système distribué considéré". Cet ordre partiel est propre à une interconnexion particulière des composants, c'est-à-dire propre au calcul particulier.

De cette compréhension des synchronisations, nous attendons également une compréhension des phénomènes de dissipation qui seraient nécessairement liés au calcul. Il s'agit ici d'une dissipation dont la cause ne serait pas technologique mais fondamentale, qui serait propre à des particularités de chaque calcul. Habituellement, le cadre des circuits synchrones entraîne implicitement nombre de particularités qui masquent des phénomènes fondamentaux : le temps est discret et global à tout le système, la synchronisation par l'horloge entraîne une dissipation systématique. Nous avons choisi de nous affranchir de toutes ces caractéristiques implicites. Mais cela conduit à des remises en cause très profondes des notions habituelles.

Les pioniers des circuits asynchrones avaient montré que les opérateurs de base n'étaient plus les opérations classiques sur les variables booléennes, mais devenaient des opérations sur les transitions de ces variables, appelés alors "événements", car la signification d'une transition de 0 vers 1 est identique à celle d'une transition de 1 vers 0. Nous avons été obligés d'aller plus loin dans cette remise en cause en considérant que l'objet de base est la "boucle", appelée ici "phase", et qui est la répétition infinie d'événements. Nous sommes alors amenés à ne considérer que des assemblages de telles phases. Ce sont ces assemblages qui fondent notre notion de synchronisation. Il pourra sembler curieux que nous n'envisagions que des rebouclages, mais ceci nous a paru nécessairement suggéré par le fait que dans les circuits asynchrones, tout événement est associé d'une façon ou d'une autre à un autre événement en retour qui constitue un "accusé de réception". Ceci est masqué dans le mode synchrone car tout signal doit se propager en un temps inférieur à la période d'horloge et cette règle rend inutile, car implicite, un tel "accusé de réception". Enlever l'horloge et son temps global conduit à enlever cette caractéristique implicite, et conduit à la nécessité de systématiser les boucles d'événements.

1.1 Mode d'exposition choisi

Ce texte fait dialoguer beaucoup de disciplines (physique, informatique, philosophie, langages, électronique, calcul distribué, circuits asynchrones, relativité, dissipation, propagation) et est motivé par de nombreuses questions et intuitions, au point qu'il nous paraît trop difficile d'exposer linéairement dans cette introduction tous les cheminements qui nous ont conduit aux conceptions élaborées ici.

Néanmoins, ce qui est décrit est simple et les raisonnements invoqués sont simples. En outre il nous paraît possible de dire au fil de l'exposé ce que pourrait nous apporter cette construction, et ainsi de faire sentir nos motivations.

Nous allons donc demander au lecteur d'accepter de nous suivre au début dans un formalisme qu'il pourra trouver curieux. Tout au long de ce par-

cours, nous ferons de multiples allusions à différents objets de différents domaines. Nous ne voyons pas d'autre moyen d'exposition et espérons qu'à la fin les choses s'éclaireront pour le lecteur.

Toutefois nous avons à prévenir à propos de quelques présupposés, de quelques remises en cause profondes de conceptions habituelles. Par exemple, en général, ceux qui décrivent des réalisations matérielles (machines, robots, etc.) supposent que la nature est telle que décrite par la physique, que les notions de temps et d'espace sont bien connues, et que l'ingénieur utilise ces connaissances comme préalables. Sont alors outils les connaissances mathématiques telles que les langages formels de l'informatique, et la définition de la notion de calcul. Nous demandons ici de supposer un autre ordre d'apparition de ces connaissances :

- Nous considérerons comme préalable à tout le reste la définition d'un langage, ici un langage particulier qui est celui des *traces* [1, 2], qui est utilisé (entre autres) dans la description d'une certaine classe de circuits électroniques sans horloge (la classe des *clockless DI* [2, 3, 4, 5, 6, 7, 8]). Ce langage permet de s'ouvrir à une structure du temps qui n'est pas un nombre mais un ordre partiel (comme d'ailleurs il en est en physique pour le temps de la relativité [9, 10, 11, 12, 13]).
- Nous chercherons à réaliser des objets qui matéralisent les phrases, les énoncés, de ce langage, mais en supposant que la structure de l'espace-temps n'est pas connue au préalable, car dépendante elle-même de la structure du langage. Pour ce faire, nous raisonnerons en faisant des analogies avec des objets connus de la physique. Ces analogies ne servent qu'à évoquer des possibilités, mais ne démontrent pas, car la justification supposerait connue la structure de l'espace-temps. La seule chose sur laquelle on s'appuie est le langage initial. L'ordre d'apparition des notions sera le suivant : nos premiers objets seront les *phases* (des boucles de symboles que nous composerons), puis cela nous permettra de définir la *dissipation logique* (d'une façon proche de [14, 15]), puis *temps et espace* viendront ensuite, en lien avec la propagation.

1.2 Outil formel : les traces

Dans ce qui suit, une *trace* est une suite de symboles (chaque symbole représente un "événement"). La trace définit un ordre partiel entre ces symboles, à l'aide d'autres symboles d'"opérateurs", d'une façon proche de Ebergen [6, 7].

Donnons des exemples de traces pour introduire ces symboles et opérateurs :

- Les symboles d'événements seront des lettres telles que a, b, c, etc.
- Le symbole ";" indique la relation d'ordre dans la trace, ordre logique

et non temporel, qu'on peut lire "est à gauche de", par exemple dans la trace :

$$...a; b; a; c; d...$$

Pour simplifier l'écriture, on peut ne pas écrire les symboles ";" :

$$...abacd...$$

- Le symbole "$\|$" (dit "weave") indique la coprésence sans relation d'ordre (les deux termes à droite et à gauche de "$\|$" doivent être présents chacun une fois et une seule), par exemple la trace :

$$...a; (b \parallel c); d...$$

indique qu'on doit avoir à la fois l'ordre $a; b; d$ et l'ordre $a; c; d$. Entre les deux symboles a et d, les deux symboles b et c apparaîssent chacun une fois et une seule.
- Le symbole "$|$" indique un choix exclusif (un et un seul des termes, celui à droite ou celui à gauche de "$|$" est présent), par exemple la trace :

$$...a; (b|c); d...$$

peut noter soit la trace $...a; b; d...$, soit la trace $...a; c; d...$.
Il s'agit d'un choix exclusif entre b et c. Nous ne nous intéressons pas à la façon dont est fait le choix, déterministe ou non, équitable ou non, etc.
- Un chiffre tel que "^{3}" en exposant indique une répétition (ici 3 fois). L'exposant est soit un nombre entier positif ou nul, soit "$*$" indiquant n'importe quel nombre entier positif ou nul, soit "∞" pour une répétition infinie, par exemple les traces :

$$[a; b]^3$$

$$[a]^\infty$$

équivalent respectivement à : $ababab$ et $...aaaaaaa...$.
- Le symbole "$\downarrow$" nommé "projection", permet de "supprimer" ou "cacher" des symboles, par exemple l'expression :

$$(a; b; a; c; b; d) \downarrow \{a, b\}$$

représente la trace :

$$a; b; a; b$$

qui est dite "projection de la trace $a; b; a; c; b; d$ sur le vocabulaire constitué des seuls symboles a et b".

Remarque sur la signification des mots employés : L'ordre défini par ";" n'est pas l'ordre temporel. Nous définirons le temps autrement, plus loin. L'ordre défini par ";" est l'ordre dans l'écriture du langage, que nous avons dit "de gauche à droite". Remarquons que cela a déjà l'inconvénient de faire croire que c'est spatial, puisque l'écriture est dans l'espace du papier. Disons simplement que c'est l'*ordre logique* et que cela n'augure rien, ni de spatial ni de temporel, pour une future implémentation matérielle.

Les objets matériels (composants) manipulés plus loin feront allusion aux circuits asynchrones DI (Delay-Insensitive) au sens de Ebergen et Udding [3, 5, 7] pour ce qui est de l'ordre partiel des symboles dans les traces. Mais le mot "Delay" est mal venu ici. En effet, pour nous, "DI" fera simplement allusion au fait que l'ordre défini par ";" est simplement un ordre logique, sans notion de "distance" (ni temporelle ni spatiale). La difficulté est liée au fait que dans les circuits asynchrones classiques, l'*ordre logique* est implémenté, réalisé, par l'*ordre temporel*. Mais ici nous voulons pouvoir distinguer ces deux relations d'ordre. Nous voulons pouvoir dessiner des circuits, constitués de composants, sans pour autant que ces composants soient matériels, disposés dans l'espace-temps. Ce seront des dessins dans l'espace du papier qui ne font que traduire graphiquement ce qui est exprimé par l'écriture des traces (ordre logique). Ensuite, nous préciserons quels composants graphiques peuvent trouver une implémentation matérielle, par exemple électronique, et ce que cela suppose sur la structure de l'espace et du temps physiques de ces composants matériels.

Pour des raisons que nous introduirons plus loin, nous ne distinguerons pas au début les "entrées" des "sorties", comme dans [16]. Nous ferons coexister des symboles tels que a, b, c, avec $a?$, $b!$, etc., que nous appellerons indifféremment *symboles* ou *variables* ou *événements*, dans l'esprit de [16], inspiré de rapprochements entre circuits asynchrones et relativité [12, 13, 17, 18]. Nous considérerons les mêmes opérateurs sur les traces que ceux utilisés classiquement, par ex. dans [7].

2 Phases, objets, calculs, dissipation

2.1 Phase

Définition : Nous définirons une **phase** comme une *trace* qui est une répétition infinie d'une trace finie. La plus simple serait :

$$[a]^\infty$$

C'est une répétition infinie du symbole a dans l'écriture de la trace. C'est la chaîne infinie de symboles (infinie à droite et à gauche) :

$$...a; a; a; a...$$

Si nous effectuons une symétrie droite-gauche de cette chaîne, nous obtenons la même chaîne.

Si nous cherchons à représenter cela, par une représentation graphique qui ne soit pas infinie, en cherchant une analogie avec un dispositif physique qui pourrait permettre une réalisation, on peut par exemple penser à la propagation dans une boucle (tel que dans la figure 1), mais alors il ne faut préciser ni le sens de rotation, ni la période, ni la fréquence, ni le lieu dans l'espace de quelque chose qui tournerait. Il n'y a pas encore de propagation car pas d'espace. Ce n'est pas quelquechose qui tourne dans l'espace en prenant du temps, ou alors c'est délocalisé, à la fois en "position" et en "instant".

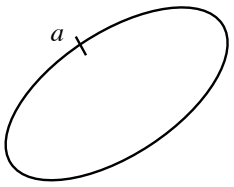


FIGURE 1 – Représentation de la phase $[a]^\infty$ comme une boucle

On peut aussi penser à une onde stationnaire, lumière entre deux miroirs (figure 2), ou corde entre deux points de fixation. Cela a l'avantage

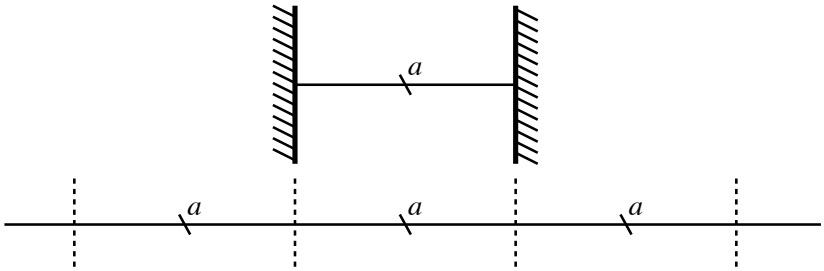


FIGURE 2 – La phase $[a]^\infty$ comme une onde stationnaire

de ne pas ajouter la notion inutile de sens de rotation ni la notion inutile de propagation. Car, dans une onde stationnaire, il n'y a ni propagation ni évolution temporelle. Mais il subsiste les notions inutiles de distance, longueur d'onde, durée, période, fréquence. C'est l'inconvénient qu'il y a à chercher dès maintenant une analogie physique, alors que nous souhaitons au contraire repousser à plus tard l'apparition des notions de matière, espace, temps. Toutefois, nous introduisons tout de même ces analogies physiques, afin de montrer les ressemblances et les dissemblances entre ces phases et

les circuits asynchrones DI. Le but est de voir à quel moment logique la physique s'introduit nécessairement.

2.2 Insertion d'un symbole dans une phase

Considérons la phase précédente, dans laquelle on insère le symbole "b", de façon ordonnée (par la relation d'ordre du symbole ";"). On obtiend une phase à deux symboles :

$$[a; b]^{\infty}$$

C'est la chaîne infinie :

$$...a; b; a; b; a...$$

Elle est aussi symétrique droite-gauche (car infinie). Chacun des deux symboles a ou b peut être centre de symétrie.

Si nous cherchons une analogie physique, nous pouvons penser à un composant A inséré dans une onde stationnaire, soit entre deux miroirs comme dans la figure 3, soit face à un seul miroir, comme dans la figure 4.

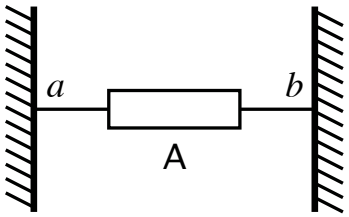


FIGURE 3 – Un composant A non-orienté, entre deux miroirs

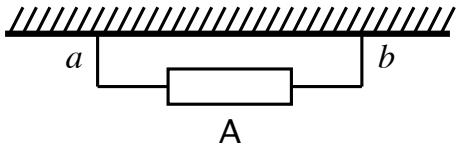


FIGURE 4 – Un composant face à un miroir

Si nous pensons plutôt en termes de boucle, nous pouvons passer de la figure 4 à la figure 5, dans laquelle deux composants symétriques A et B dialoguent à travers une interface I qui se substitue au miroir. Ici, de même que dans la figure 1, ce serait préciser trop de choses que de définir un sens de rotation ou une propagation. Chacun des composants A et B joue le même rôle que le composant des figures 3 ou 4. Pourtant cette figure 5 commence

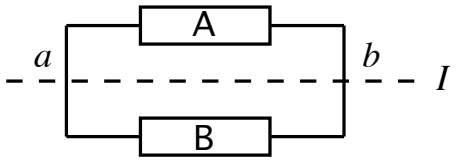


FIGURE 5 – Deux composants et une interface

à ressembler à une figure composée de deux composants asynchrones DI qui supposerait une propagation d'événement.

Par la suite, nous aurons besoin, dans certains cas, de pousser plus loin cette ressemblance. A et B seront alors des composants, a et b seront des bornes de connexion entre composants. Nous distinguerons deux étapes : d'abord des composants "non-orientés", ensuite des composants "orientés". C'est seulement dans cette dernière étape que les bornes seront nommées "entrée" ou "sortie", et écrites respectivement $a?$ et $a!$.

Dès que l'on parle d'*interface* entre composants, nous avons envie de couper le long de cette ligne d'interface, et ainsi d'*isoler* chaque composant. À partir de ce moment, la trace $[a; b]^\infty$ prend d'autres significations. On peut la considérer comme *le langage de ce qui transite à travers l'interface*. On peut aussi la considérer comme la *fonction*, ou la *spécification* de chacun des composants qui est connecté à cette interface, pris de façon isolée, en *boucle ouverte*.

Il y a une importante difficulté qui surgit alors, et qu'il ne faut pas laisser dans l'ombre. *Isoler* un composant, c'est-à-dire en parler comme d'un objet *autonome*, est une situation très différente de ce qui se passe quand on ne considère que la globalité de la boucle. En effet, un "composant" est censé se comporter de la même façon dans toutes les "compositions" où il intervient. C'est-à-dire que sa *fonction*, ce qu'il *fait*, son *comportement* est indépendant de ce qui se trouve de l'autre côté de l'interface par laquelle il est connecté. Cet "au-delà de l'interface" est alors appelé son "environnement". La réalisation physique du composant doit être *compatible* avec une grande variété d'environnements possibles, voire une infinité. Or ceci est une contrainte très forte lorsqu'il s'agit d'envisager une réalisation physique, car les environnements peuvent être très variés. Pour les montages électroniques auxquels nous sommes habitués, nous avons l'habitude de ces contraintes qui se manifestent sous la forme de questions d'*immunité au bruit*, d'*amplification*, de *stabilité des mémoires*, et ceci conduit à ne considérer que des composants qui sont *orientés* car ils comportent une amplification, cette dernière étant également associée à une *différence d'impédance* entre l'*entrée* et la *sortie*. Tout cela est également lié à la question de la *consommation d'énergie* et à la *dissipation thermique*.

En revanche, ces problèmes ne s'introduisent pas nécessairement lorsque

nous n'envisageons les montages que de façon globale, tel que dans une boucle ou une onde stationnaire, sans "décomposer" en "composants". Par exemple, les représentations globales et non-orientées des figures 1 à 4 permettent d'envisager des réalisations non amplifiées et non-dissipatives (par exemple cavité résonnante sans pertes). *Dans ce qui suit, nous aurons le souci de dégager, de mettre en évidence, les endroits ou le caractère "orienté" des composants intervient nécessairement.*

Dans la figure 5, il y a une particularisation des deux composants par le dessin, mais tant qu'on ne les sépare pas, tant qu'on n'envisage que la description globale, alors il n'est pas nécessaire d'orienter les composants, et une réalisation physique globale de la figure 5 pourra mimer ce qui est représenté figure 4. Ceci est particulier à cet exemple dans lequel les deux composants symétriques A et B de la figure 5 peuvent être identiques.

Mais dans le cas général, la *symétrie* ne sera pas l'*identité*. Par la suite, nous dirons que les deux composants qui se font face dans une symétrie sont "**Anti-**" l'un de l'autre et nous parlerons d'**anti-composants**.

Les deux composants de la figure 5 qui sont identiques entre eux seront nommés chacun "**Wire non-orienté**" (Note : nous utiliserons un terme anglais comme nom d'objet chaque fois qu'une traduction aurait eu l'inconvénient de faire perdre une allusion à une publication connue, ici [7] par exemple). Nous dirons donc que le Wire non-orienté est son propre anti-composant, ou bien que l'"Anti-Wire non-orienté" est un "Wire non-orienté". Sa fonction, c'est-à-dire sa spécification, est donnée par :

$$[a; b]^{\infty}$$

Ici, cette expression est donc devenue la *spécification* d'un composant isolé (le "Wire non-orienté"), a priori distincte de la définition de la *phase* qui était l'objet global.

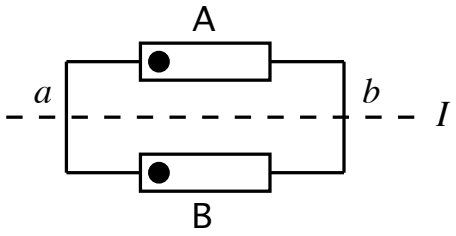


FIGURE 6 – Connection de deux "Wires non-orientés"

Mais nous voyons là que si nous voulons connecter A isolé à B isolé, nous devons faire correspondre a de A à a de B et b de A à b de B car dans le cas général, les deux extrémités d'un composant ne sont pas forcément permutables, même si le composant n'est pas orienté. Nous allons donc, par convention, marquer sur le schéma, à l'aide d'un point noir, le symbole

qui correspond à celui qui est écrit en premier dans la spécification. La figure 5 devient alors la figure 6. Dans cette figure, il n'y a toujours pas de notion de rotation, de circulation, de propagation, les deux composants sont identiques et collaborent pour mimer le composant de la figure 4. Le *point noir* ne marque pas une *initialisation*, puisque les chaînes $[a;b]^\infty$ sont infinies à droite et à gauche, mais ce point indique une *synchronisation* entre les composants A et B :

Wire A : $...a;b;a;b...$

Wire B : $...a;b;a;b...$

Il est clair que la fonction d'un composant ne dépend pas de son nom ni des noms de ses bornes de connexion. Elle ne dépend que de la forme de sa spécification. Il nous sera donc possible de renommer les composants et les bornes, à condition de choisir des nouveaux noms qui ne créent pas d'ambiguïtés avec des noms existants. Par convention, une identité de noms de bornes représentera une connexion entre ces bornes. Par exemple, on pourra connecter les deux Wires non-orientés $[a;b]^\infty$ et $[c;d]^\infty$ pour former une phase, après avoir renommé les bornes du second en a et b.

2.3 Objets orientés

Comme nous l'avons annoncé, l'étape suivante consiste à passer à des composants orientés, dont nous aurons besoin par la suite. Pour notre phase élémentaire, il suffit de transformer la figure 6 en imposant un sens de rotation. Nous obtenons alors la figure 7 (dans laquelle le sens de propagation de chaque composant est indiqué par les symboles “>” et “<”, ici dans la forme du contour). Les composants symétriques ne sont plus identiques.

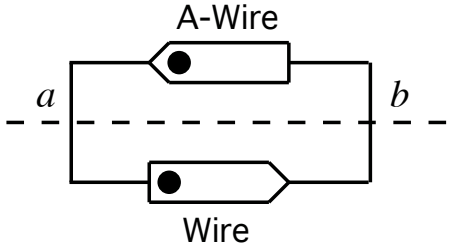


FIGURE 7 – Wire orienté et Anti-Wire orienté

Leurs implémentations sont plus conformes à ce dont nous avons l'habitude en électronique. Chacun comporte une borne d'*entrée* et une borne de *sortie*. Entre l'*entrée* et la *sortie* du composant (c'est-à-dire à l'intérieur du composant), la propagation est assurée par un montage *amplificateur* qui apporte en outre une *immunité au bruit*.

– La phase s'écrit encore :

$$[a;b]^\infty$$

– L'objet dessiné en bas, dénommé "Wire orienté", a pour spécification :

$$[a?; b!]^{\infty}$$

– L'objet dessiné en haut, dénommé "Anti-Wire orienté", a pour spécification :

$$[a!; b?]^{\infty}$$

(Dans une spécification d'objet orienté, le "?" désigne une *entrée*, le "!" désigne une *sortie*.)

Le *point* indique comme précédemment le calage, la synchronisation des chaînes, mais ici il apporte en outre la correspondance entre la sortie d'un composant et l'entrée de l'autre :

Anti-Wire orienté : $...a!; b?; a!; b?...$
Wire orienté : $...a?; b!; a?; b!...$

$a!$ est connecté avec $a?$, ce que nous représentons ici par l'alignement vertical des traces. Nous remarquons que le Wire a autant besoin de synchronisation que le Anti-Wire, puisqu'ils doivent être connectés l'un à l'autre. (Remarquons que ceci est souvent passé sous silence par les autres auteurs, par exemple chez Ebergen [7] où l'Inverted-Wire a un rôle lors de l'initialisation, mais non le Wire. Ceci n'est pas étonnant car cette nécessité de synchronisation du Wire n'apparaît que lorsque nous introduisons la symétrie des composants comme nous le faisons ici avec les phases.)

Nous pouvons remarquer que l'expression écrite qui dénote la phase, tout comme les spécifications des composants, sont encore des chaînes symétriques droite-gauche. Cette propriété de symétrie est dissociée de la question du sens de rotation. Cela nous indique ici que nous ne sommes pas obligés d'implémenter cette phase avec des objets orientés.

Relations avec les définitions des autres auteurs

Nous pouvons remarquer que notre "Anti-Wire" ressemble fortement à l'"Inverted-Wire" de Ebergen dans [6, 7]. Il en diffère cependant : les composants de Ebergen sont *initialisés*. C'est-à-dire que pour eux, il y a un commencement du temps. Les chaînes peuvent être infinies à droite, non à gauche. C'est ce qui nécessite le Inverted-Wire chez Ebergen, car ce composant, juste après l'initialisation, commence par émettre un événement en sortie. Après cela, cet Inverted-Wire ne se distingue en rien d'un Wire.

Puisque nos chaînes sont infinies, nous n'avons pas besoin d'initialisation. Mais nous avons tout-de-même besoin de la notion de "Inverted". En effet, dès que nous avons des composants orientés, il nous faut faire une différence entre l'état d'un composant qui attend un événement en entrée, et l'état d'un composant lorsqu'il se prépare à émettre un événement en sortie. Dans le premier cas le composant est passif et attend une initiative de l'environnement (entrée), alors que dans le second cas le composant doit être actif car l'environnement attend une initiative du composant (sortie). Et bien entendu, cela est important lorsqu'il s'agit de connecter des composants

car il doivent être synchronisés. Par convention, nous choisirons d'appeler "**Inverted**" : *un composant dont le crochet de spécification commence par un symbole de sortie*. Ici, ce ne sera donc pas une question d'initialisation liée à un "*reset*", mais ce sera une question de *calage*, de *synchronisation* entre les composants.

Avec cette définition, notre "Anti-Wire orienté" est donc également un "Inverted-Wire orienté". Ceci est un cas particulier dû à la simplicité de la fonction "Wire". En général, le qualificatif "Anti" ne sera pas équivalent à "Inverted". "Anti" se rapporte à la symétrie, alors que "Inverted" se rapporte au calage de la synchronisation.

Chez Van de Snepscheut [2], il y a les notions de "Undirected Trace" et "Directed Trace" qui correspondent à notre distinction entre les traces d'objets non-orientés et les traces d'objets orientés.

Nous pouvons remarquer que pour les objets non-orientés, la symétrie "Anti" est l'identité. Pour les objets orientés, cette symétrie n'est pas l'identité et change *au moins* le sens de propagation.

Chez la plupart des autres auteurs, tous les composants sont d'emblée considérés comme des objets matériels insérés dans l'espace et le temps. Ainsi, il n'est considéré que des objets orientés, et dans ce cas la relation d'*ordre (logique)* dans les traces est la même que la relation d'*ordre temporel*. Pour nous, nous ne considérerons cette relation avec le temps physique que dans le cas où un montage apparaîtra comme nécessairement orienté. Dans tous les autres cas, où nous pourrons maintenir une description non-orientée, nous envisagerons d'autres types de réalisations, dans lesquelles l'ordre des symboles ne sera pas nécessairement l'ordre temporel. Dans ce cas, les composants ne seront pas forcément propagatifs. Ils ne nous servirons pas uniquement dans un but de réalisation, mais ils nous serviront également dans ce cadre théorique pour voir où la matérialité et la propagation s'introduisent par nécessité.

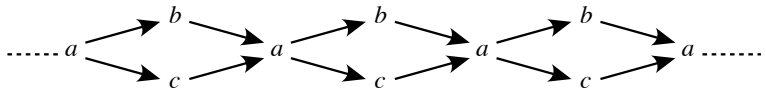
Notre démarche peut d'une certaine manière être mise en parallèle avec celle de [19], où l'on évoque l'image miroir des composants en relation avec un renversement du temps. Mais dans [19] tous les composants sont renversables, alors qu'au contraire pour nous la présence de composants non renversables sera un critère montrant la nécessité d'introduction de la dissipation, et donc la nécessité de l'introduction d'un temps physique, temps dont la structure dépendra donc du langage qui a conduit à cette dissipation.

2.4 Introduction du symbole "‖", nommé "Weave"

Soit la phase :

$$[a; (b \parallel c)]^{\infty}$$

qui est la chaîne suivante (les flèches entre les symboles dénotent la relation d'ordre ";") :



Entre deux symboles a successifs, les deux symboles b et c doivent apparaître chacun une fois et une seule. Notons que cette chaîne est symétrique droite-gauche car chaque symbole a est centre de symétrie, ainsi que chaque couple $(b \parallel c)$.

Graphiquement, en terme d'objets non-orientés, nous pouvons la représenter par les figures 8 (avec un miroir) ou 9 (avec deux objets symétriques).

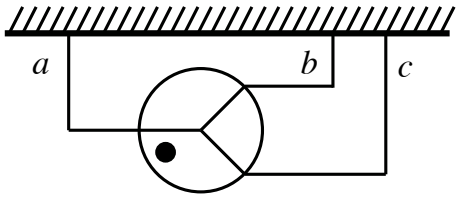


FIGURE 8 – Weave non-orienté (avec un miroir)

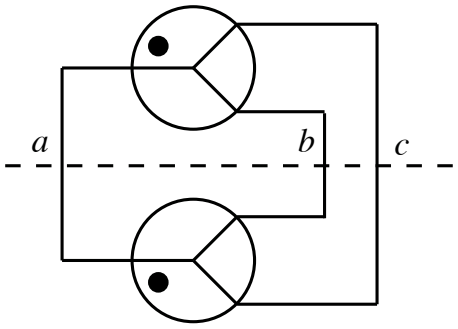
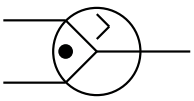


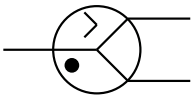
FIGURE 9 – Weave non-orienté (avec deux objets symétriques)

Nous ne donnerons pas d'autre nom que "weave" à ces objets (composants non-orientés), mais nous nommerons spécifiquement ensuite les composants orientés, en cherchant à nous rapprocher des noms qui évoquent les objets similaires des autres auteurs.

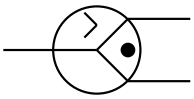
Si nous cherchons à introduire des composants orientés, nous devons en définir quatre (le symbole ">" indique le sens de propagation) :



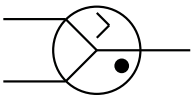
$[(b? \parallel c?); a!]^\infty$ habituellement nommé "C-Muller"



$[a?; (b! \parallel c!)]^\infty$ habituellement nommé "Fork"



$[(b! \parallel c!); a?]^\infty$ "Inverted-Fork" ou "Anti-C-Muller"



$[a!; (b? \parallel c?)]^\infty$ "Inverted-C-Muller ou "Anti-Fork"

On remarque qu'on passe de "C-Muller" à "Fork" et inversement en appliquant une fois chacune des deux opérations "Anti" et "Inverted". "Anti" consiste à remplacer les entrées par des sorties *et vice versa*. "Inverted" consiste à inverser la droite et la gauche dans le crochet (pour ce cas particulier simple).

Remarque : Le Fork introduit ici ne peut pas être qualifié par rapport à un "isochronisme", qui serait une égalité des temps de propagation dans les deux branches de la fourche, puisqu'il n'y a pas de notion de temps, ni à plus forte raison de métrique temporelle. Il y a uniquement une relation d'ordre partiel entre entrée et sorties.

Avec ces 4 composants orientés, nous pouvons construire l'association C-Muller et Anti-C-Muller :

$$[(b \parallel c); a]^\infty$$

ainsi que l'association Fork et Anti-Fork :

$$[a; (b \parallel c)]^\infty$$

Les chaînes construites par ces deux associations sont les mêmes. Elles peuvent être représentées par un dessin proche de celui de la figure 9, mais en restreignant à chaque fois à un seul sens de rotation dans la boucle (puisqu'il s'agit maintenant de composants orientés).

Toutefois, puisque le sens de rotation ne se voit pas sur l'écriture de la phase, il n'est pas nécessaire de recourir à des composants orientés pour implémenter cette phase. On pourrait donc envisager une réalisation de cette phase à l'aide de "Weaves non-orientés".

2.5 Introduction d'une alternance de symboles

Considérons la phase :

$$[a; b; c; b]^{\infty}$$

qui est la chaîne :

$$...a; b; c; b; a; b; c; b; a...$$

Les symboles a et c alternent pour séparer les symboles b. Notons que cette chaîne est symétrique droite-gauche car chaque symbole a ou c est un centre de symétrie (mais non les symboles b).

Graphiquement, en terme d'objets non-orientés, nous pouvons la représenter par les figures 10 (avec un miroir) ou 11 (avec deux objets symétriques). Le rond blanc indique le calage de la borne a, alors que le point noir indique la synchronisation avec l'"Anti-" comme précédemment. Nous ne donnerons

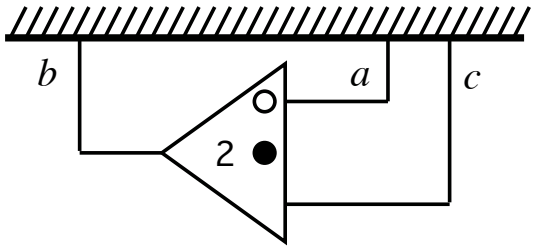


FIGURE 10 – Alternance non-orientée (avec un miroir)

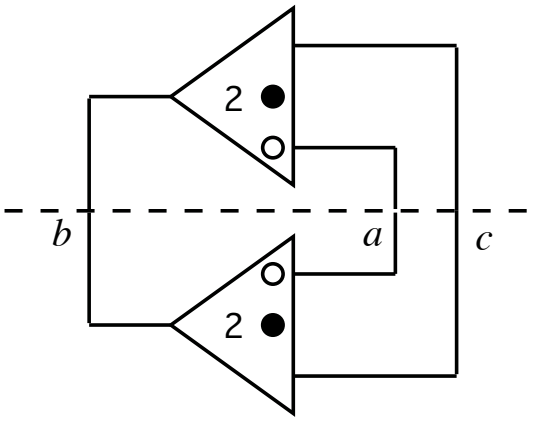
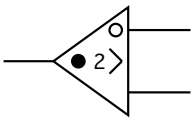


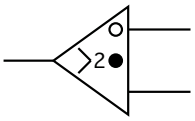
FIGURE 11 – Alternance non-orientée (avec deux objets symétriques)

pas de noms à ces objets, mais nous nommerons uniquement les composants orientés, en cherchant à nous rapprocher des noms qui évoquent les objets des autres auteurs (de la famille du "Toggle" ou diviseur par 2).

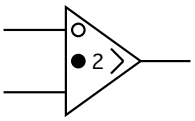
Si nous cherchons à introduire des composants orientés, nous devons en définir quatre. Nous serons conduit à inventer le nouveau nom de "**Elggot**" pour symétriser le" Toggle".



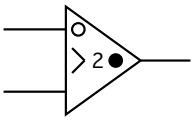
$[b?; a!; b?; c!]^{\infty}$ habituellement appelé "Toggle"



$[a!; b?; c!; b?]^{\infty}$ "Inverted Toggle"



$[a?; b!; c?; b!]^{\infty}$ "Anti-Inverted-Toggle" ou "Elggot"



$[b!; a?; b!; c?]^{\infty}$ "Inverted Elggot" ou "Anti-Toggle"

Le symbole ">" indique le sens de propagation.

Remarquons que si nous souhaitons que les composants "Inverted" soient représentés par des crochets de traces qui commencent par une sortie, nous sommes amenés à décaler les symboles dans les crochets. Mais chacun s'apparie bien avec son "Anti".

Remarquons que les deux derniers composants (Elggot et Inverted-Elggot) ne fonctionnent correctement (ne réalisent leur spécification) que si l'environnement qui leur fournit les entrées respectent la bonne alternance de la spécification. Ceci impose une contrainte sur l'environnement qui est un peu inhabituelle. C'est sans doute pour cette raison que les auteurs précédents n'ont pas introduit cette primitive "Elggot". Mais pour nous, l'Elggot est au Toggle ce que le Fork est au C-Muller, son introduction est tout aussi nécessaire. (Le Fork paraît tellement plus "naturel" !) Le fait d'imposer une contrainte sur l'environnement n'est pas une nouveauté, il y en a pour tous les composants, mais celle-ci est juste un peu plus contraignante dans sa forme que le sont les contraintes habituelles.

2.6 Généralisation de l'alternance de symboles : division par N

Nous pouvons généraliser la division par 2 à la division par N. Considérons la phase :

$$\left[[a;b]^{N-1};c;b\right]^{\infty}$$

qui est la généralisation du Toggle à la division par N. Nous l'appellerons "N-Div".

Notons que cette chaîne est symétrique droite-gauche, les symboles c étant centres de symétrie.

Graphiquement, en terme d'objets non-orientés, nous pouvons la représenter par les figures 10 ou 11 dans lesquelles le chiffre "2" serait remplacé par "N".

Si nous cherchons à introduire des composants orientés, nous devons en définir quatre :

- "N-Div"
- "Inverted-N-Div"
- "Anti-Inverted-N-Div" (ou "N-Elggot")
- "Anti-N-Div

Nous ne détaillerons pas davantage car cela s'étend aisément à partir des composants de la famille des Toggles.

Il faudrait également envisager, non 4, mais $4N$ objets orientés, suivant les différents modes de calage :

$$\left[[a;b]^{N-i-1};c;b;[a;b]^{i}\right]^{\infty}$$

$$0 \leq i \leq N-1$$

Mais ces traces s'obtiennent simplement par rotation des traces initiales, et sont les mêmes puisque nous sommes dans un contexte où les traces sont infinies à droite et à gauche. Nous ne détaillerons pas davantage la différence sur les implémentations des composants.

2.7 Choix exclusif de symboles

Considérons la phase suivante, dans laquelle pour chaque parenthèse un seul des deux symboles b ou c apparaît une fois et une seule. Il s'agit d'un choix exclusif entre b et c.

$$[(b|c);a]^{\infty}$$

On pourrait écrire cette chaîne ainsi, si cette écriture n'était pas trompeuse :

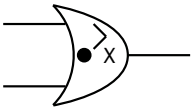
$$...a;(b|c);a;(b|c);a...$$

Cette écriture générique est trompeuse car elle semble symétrique droite-gauche, alors qu'un exemple particulier d'une telle chaîne tel que :

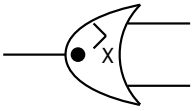
$$...a; b; a; b; a; c...$$

n'est en général pas symétrique droite-gauche. (Remarquons que *c'est la première fois* que nous rencontrons une *chaîne non symétrique.*)

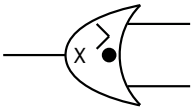
En raison de cette dissymétrie, cherchons, d'emblée cette fois, à introduire des composants *orientés.* Nous devons en définir quatre, et nous introduisons le terme de "**Décideur**" :



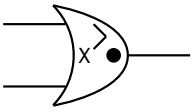
$[(b?|c?); a!]^\infty$ "Merge"



$[a?; (b!|c!)]^\infty$ "Décideur"



$[(b!|c!); a?]^\infty$ "Inverted-Décideur" ou "Anti-Merge"



$[a!; (b?|c?)]^\infty$ "Inverted-Merge" ou "Anti-Décideur"

Le "X" sur le schéma indique le caractère exclusif, le point noir indique la synchronisation comme précédemment, le symbole ">" indique le sens de propagation.

Le Merge est classique.

Le Décideur est de la famille des *arbitres* puisqu'il a à faire un choix, c'est-à-dire à prendre une décision entre deux sorties exclusives. Nous voyons ici que la fonction du Merge peut être considérée comme symétrique de celle d'un arbitre. En effet, le Décideur *crée une information* en faisant un choix, alors que le Merge *oublie ou annihile cette information* en rendant indifférencié l'événement qui le traverse depuis une entrée vers sa sortie.

Notons que nous ne faisons aucune supposition sur la nature du choix du Décideur. Est-il déterministe ou non ? Est-il aléatoire ou non ? Est-il équitable ou non ? Toutes ces questions sont du ressort de l'implémentation, non du langage lui-même. Le Décideur n'a ici pas d'autre caractéristique que d'être le symétrique du Merge : là où le Merge est capable d'*annihiler* une information, le Décideur doit être capable d'en *créer* une.

La pratique de la réalisation des arbitres électroniques nous apprend qu'un choix peut être implémenté avec un montage qui, à partir d'un équilibre métastable, amplifie un déséquilibre issu d'un bruit, thermique ou autre. Il est vraisemblable que si le bruit est thermique, le choix sera aléatoire, alors que si le bruit est corrélé à autre chose, le choix sera déterminé par cette autre chose. Une caractéristique importante de ces arbitres est que *le temps de décision est non-borné*, c'est-à-dire que, quelque soit le délai qu'on peut lui allouer pour sa prise de décision, il est possible qu'il prenne, pour décider, un temps supérieur à ce délai imparti [20, 21, 22, 23, 24, 25].

Est-il possible de réaliser les objets *non-orientés* représentés par les figures 12 (avec un miroir) ou 13 (avec deux objets symétriques) ?

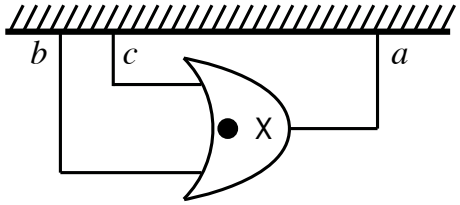


FIGURE 12 – Choix exclusif non-orienté ? (avec un miroir)

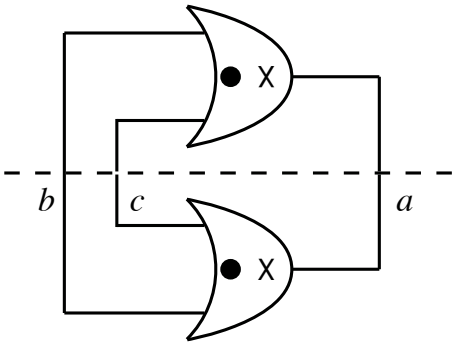


FIGURE 13 – Choix exclusif non-orienté ? (avec 2 objets symétriques)

Nous ne savons pas répondre à cette question, mais il nous semble que les fonctions de *création* et d'*annihilation* d'information sont trop différentes pour qu'un même objet puisse les prendre en charge par une simple symétrie. En outre, il nous semble que ces fonctions doivent nécessairement *impliquer de l'énergie en raison de leur irréversibilité logique*. C'est pourquoi nous définirons ci-dessous une notion de dissipation d'une façon telle qu'elle apparaisse comme liée au manque de symétrie droite-gauche des chaînes de symboles. Certes, tout ceci n'est que conjecture et proposition. Aussi, notre notion de dissipation (purement logique) ne sera pas forcément reliée à une dissipation physique.

2.8 Autres définitions : calcul, dissipation

Dans tout ce que nous avons vu précédemment, toutes les chaînes sont des répétitions infinies. En outre, elles sont presque toutes symétriques droite-gauche, si on excepte le cas du choix exclusif que nous venons de traîter.

Toutes les chaînes symétriques droite-gauche ont été montrées comme étant supposées pouvoir être implémentées par un montage non-dissipatif, soit par une onde stationnaire face à un miroir, soit par le face-à-face d'un objet non-orienté avec son Anti-objet, de part et d'autre d'une interface qui ne fait que simuler la place d'un miroir.

Cette dernière présentation, avec une symétrie de part et d'autre d'une interface, a permis d'introduire, à l'étape suivante, des composants orientés, qui semblent nécessaires pour les chaînes non-symétriques. De ces dernières nous n'avons montré qu'un seul exemple, celui du choix exclusif.

Cette présentation en composants orientés nous rappelle les composants des autres auteurs qui ont traîté de circuits asynchrones DI, pour lesquels chaque composant est alors un composant électronique (donc matériel) fonctionnant dans l'espace-temps. Le temps est alors ce qui est assimilable à l'ordre logique de succession des symboles.

Notre proposition est que cette question d'orientation des composants est liée à la question de la dissipation.

Dans ce qui suit, nous allons introduire des chaînes plus complexes, en combinant les opérateurs de base ";", "||" et "|". Nous serons amenés à considérer en général des interfaces qui ne sont pas des symétries (non-assimilables au miroir).

Toutefois, nous allons chercher à maintenir la possibilité des implémentations non-orientées, afin de garder à l'esprit que des chaînes complexes peuvent être symétriques droite-gauche, et qu'elles pourraient permettre des implémentations non-dissipatives. Dans ces cas, nous dirons que le temps n'existe pas (ou bien le temps n'aura pas la même définition !) Ou pour dire autrement, les composants logiques ne seront pas implémentés par des composants matériels individualisables à travers lesquels se propage quelque chose. La relation entre la description logique et l'implémentation physique sera d'une autre nature.

Définition : Un **calcul** est un *ensemble de phases synchronisées.*

Dans les exemples de calculs que nous avons vus jusqu'ici, il n'y a eu en général qu'une seule phase, sauf dans le cas du *weave* $[a;(b \parallel c)]^\infty$ qui peut être vu comme la collaboration synchronisée des deux phases $[a;b]^\infty$ et $[a;c]^\infty$. La synchronisation est ici définie comme la mise en commun du symbole a.

Définitions : Nous appellerons **non renversable** ou **dissipative** une phase qui n'est pas symétrique droite-gauche. Nous considérerons (comme une conjecture) qu'elle n'est implémentable qu'à l'aide de composants orien-

tés dissipatifs. Alors qu'une phase symétrique droite-gauche *pourrait éventuellement* trouver une implémentation non-orientée non-dissipative (en plus des implémentations orientées dissipatives).

Voici deux exemples de traces dissipatives :

1. $[a; b; c]^\infty$
2. $[(b|c); a]^\infty$

Dans l'exemple 1, considérons la *projection* de la trace $[a; b; c]^\infty$ sur l'ensemble constitué des symboles a et b, c'est-à-dire ce qui reste de cette trace lorsque nous avons enlevé les symboles autres que a ou b. Nous noterons cette projection :

$$[a; b; c]^\infty \downarrow \{a, b\}$$

Nous dirons qu'ainsi nous "cachons" la variable c. Le résultat de la projection est :

$$[a; b]^\infty$$

C'est une trace non-dissipative.

Cet exemple nous montre qu'on peut *passer d'une phase dissipative à une phase non-dissipative en cachant des symboles.*

L'opération inverse (le contraire de "cacher") est ce qui est obtenu lorsque nous faisons une *implémentation* ou *synthèse.* En effet, nous sommes alors amenés à *décomposer* une opération, c'est-à-dire à implémenter une opération par un *assemblage de composants.* Ceci introduit des *variables intermédiaires* qui peuvent rompre la symétrie (ce sont en fait des "*connexions supplémentaires*" nécessaires pour connecter les composants).

Ainsi, lorsqu'un calcul est donné globalement par une phase non-dissipative, *son implémentation* (synthèse) par décomposition *introduit en général de la dissipation* (car l'implémentation introduit des variables internes qui rompent la symétrie).

Définition : Nous appellerons **dissipation logique** la *quantité qui mesure le nombre de symboles à cacher* dans une trace dissipative pour obtenir une trace non-dissipative. Pour une phase (qui est une chaîne infinie), afin que ce nombre ne soit pas infini, nous le rapporterons à un cycle de la phase.

Dans l'exemple $[a; b; c]^\infty$ nous dirons que la dissipation logique est de "un symbole par cycle".

Pour $[(b|c); a]^\infty$ la dissipation est de "deux symboles par cycle" (les symboles b et c).

Définition : Lorsqu'il y a *dissipation logique*, nous dirons qu'il y a du **temps**. Dans ce cas, comme la trace n'est pas identique à sa symétrique droite-gauche, l'implémentation orientée n'est pas identique à son image miroir (l'autre sens de rotation). Dans une telle implémentation à l'aide d'objets orientés, nous pouvons dire que le temps avance d'une unité à chaque instance de symbole liée à une dissipation (symbole qu'il faudrait cacher

pour obtenir une phase non-dissipative). *L'unité de durée est la même que l'unité de dissipation logique, c'est "le symbole"*.

Le temps apparaît alors comme une conséquence nécessaire de ce que nous avons appelé dissipation, qui est définie à partir de la dissymétrie droite-gauche.

Ceci nous conduit, *pour les phases dissipatives uniquement*, à nous représenter une phase réalisée à partir d'*objets orientés* comme *un circuit dans lequel circule un jeton*. Car en effet, le temps qui avance peut être lié à la propagation d'un jeton qui avance, puisqu'il y a un "avant" et un "après" l'instant du passage de l'occurence du symbole dissipatif. Mais ainsi les seuls instants où le jeton semble localisé sont les occurences de symboles dissipatifs. En dehors de ces instants, il n'y a pas de raison de considérer qu'il y a un jeton, et il n'y a pas de raison de parler de localisation.

Cette représentation sous forme d'un jeton qui avance ne correspond à rien pour les phases non-dissipatives puisqu'il n'y a pas de raison de favoriser un sens de rotation, ni de considérer une localisation de quelque chose. Dans ces cas, nous dirons qu'il n'y a pas de temps, ni matérialité d'un jeton.

Nous ne parlerons même pas de "temps cyclique", car rien ne distingue alors une itération de la suivante. Il y a certes un cycle, mais pas de propagation, donc "rien qui repasse au même lieu en un temps ultérieur".

Pour les traces dissipatives, la dissymétrie droite-gauche conduit à choisir un sens de parcours de la trace, ce qui choisit un sens du temps et un sens de rotation dans les objets orientés. Si on implémente cela physiquement par des circuits électroniques, on fait alors le choix de ce qui est "entrée" et de ce qui est "sortie" et cela conduit à un choix d'impédance et donc à une amplification qui conduira à de la dissipation physique. C'est ainsi que nous voyons le lien entre, d'une part, notre dissipation logique définie d'après un langage, et d'autre part, la dissipation physique. Entre les deux, il nous a fallu passer par le temps, la propagation, la synchronisation, l'espace, la matière, l'énergie. Notons que l'introduction du temps vient après celle de la dissipation logique.

3 Décomposition et composition

3.1 Traces obtenues par combinaison d'opérateurs logiques : décomposition d'objet

Considérons l'exemple de la phase suivante :

$$[a; c; (d \parallel e); c]^{\infty}$$

Remarquons qu'elle est symétrique droite-gauche, le symbole "a" ainsi que "$(d \parallel e)$" sont centres de symétrie.

Graphiquement, en termes d'objets non-orientés, nous pouvons par exemple la représenter par les figures 14 (avec un miroir) ou 15 (avec deux objets symétriques). Nous nommerons T (pour "Truc") l'opérateur correspondant.

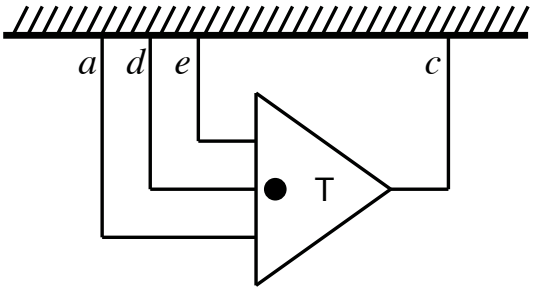


FIGURE 14 – Truc non-orienté (avec un miroir)

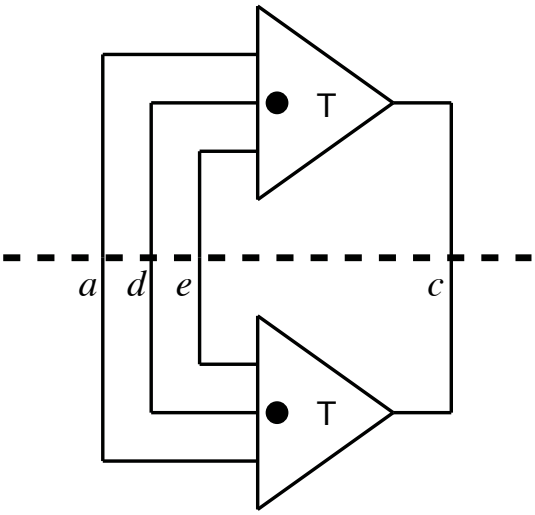


FIGURE 15 – Trucs non-orientés

Si nous cherchons à introduire des objets orientés, nous obtiendrons les quatre objets : Truc, Inverted-Truc, Anti-Truc et Anti-Inverted-Truc.

Mais nous pouvons aussi tenter de considérer une **décomposition** faisant apparaître des composants déjà rencontrés, comme par exemple dans la figure 16 avec des composants non-orientés.

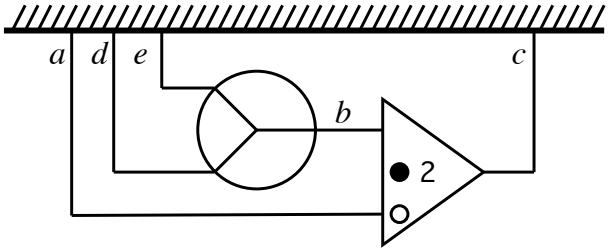


FIGURE 16 – Décomposition de T en composants non-orientés

La décomposition fait apparaître le nouveau symbole b et la chaîne complète devient :

$$[a; c; b; (d \parallel e); b; c]^{\infty}$$

Remarquons que cette insertion du symbole b ne rompt pas la symétrie droite-gauche de la chaîne et ne semble donc pas impliquer l'apparition d'une dissipation.

En revanche, si nous réalisons une décomposition avec des composants orientés comme dans la figure 17, alors la chaîne devient :

$$[a; c; b'; (d \parallel e); b; c]^{\infty}$$

Cette chaîne n'est plus symétrique droite-gauche et est donc dissipative.

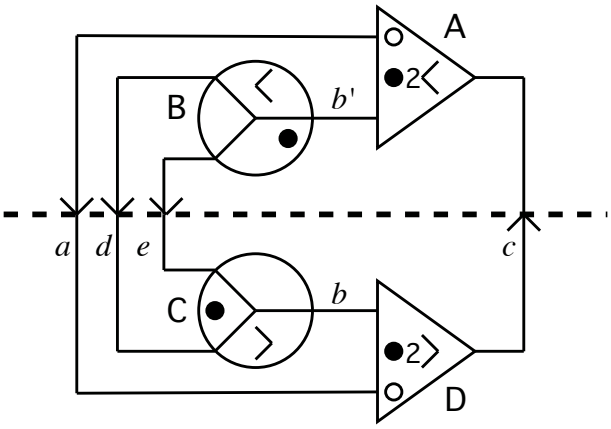


FIGURE 17 – Décomposition de Truc en composants orientés

Nous avons ainsi fait certaines remarques que nous chercherons à généraliser : la décomposition ne conduit pas nécessairement à davantage de dissipation, en particulier si le montage, une fois décomposé, est encore constitué de composants non-orientés. Mais si le montage, une fois décomposé, comporte des composants orientés, alors il y a introduction de nouveaux symboles qui conduisent en général à de la dissipation.

3.2 Multiplication des interfaces

Nous venons d'aborder la décomposition, ou synthèse d'un opérateur (Truc), et avons vu que cela entraîne l'ajoût de nouveaux symboles, ce qui peut ajouter de la dissipation.

Avec la décomposition en composants orientés de la figure 17, nous nous retrouvons dans un contexte bien connu, celui des circuits asynchrones DI. Nous n'avons mentionné sur cette figure qu'une seule interface (ligne pointillée), qui est celle issue de la symétrie originelle, mais il est clair que se présentent maintenant à nous d'autres interfaces, puisqu'il y a d'autres façons de découper en blocs de composants, qu'on peut décrire aussi comme d'autres façons de projeter la chaîne initiale.

Par exemple, la projection sur les symboles-bornes du composant A sera :

$$[a; c; b'; c]^\infty$$

qui fait penser à la famille des Toggles. Si nous privilégions l'étude du composant A plutôt que l'étude de son "Anti" (son environnement), alors cela précisera l'orientation des bornes d'entrée-sortie :

$$[a!; c?; b'!; c?]^\infty$$

qui est la spécification d'un Inverted-Toggle (pour A), associé à un environnement qui se comporte donc comme un Anti-Inverted-Toggle (ou Elggot).

Si on fait une telle projection pour les 3 autres composants, nous trouverons que B est un Fork (associé à un environnement qui se comporte comme un Anti-Fork) :

$$[b'?; (d! \parallel e!)]^\infty$$

que C est un C-Muller (associé à un environnement qui se comporte comme un Anti-C-Muller) :

$$[(d? \parallel e?); b!]^\infty$$

et que D est un Elggot (associé à un environnement qui se comporte comme un Anti-Elgott) :

$$[a?; c!; b?; c!]^\infty$$

Remarquons qu'en effectuant ces projections, nous ne trouvons qu'un seul composant qui est "Inverted", c'est celui qui comprend parmi ses sorties

le symbole qui est le premier dans l'écriture de la phase. Ceci justifie la position des *points noirs* que nous avons mis figure 17.

Il y a d'autres interfaces possibles (d'autres façons de couper), par exemple la projection sur $\{b', b\}$:

$$[b'; b]^\infty$$

qui fait apparaître le couple (Wire, Anti-Wire), le Wire étant : $[b'?; b!]^\infty$. Le Wire est composé de l'assemblage "Fork + C-Muller", alors que l'Anti-Wire est composé de l'assemblage "Inverted-Toggle + Elgott".

Nous voyons ainsi *émerger une nouvelle notion de composant*, c'est ce qui apparaît *à chaque interface*, c'est-à-dire *pour chaque projection* : à chaque fois un *couple* de part et d'autre de cette interface, couple *composant / Anti-composant*. Dans certains cas, un des deux éléments de ce couple correspond à la même notion de composant que celle dont nous avons l'habitude, c'est le cas ici pour les composants que nous avons nommés A, B, C et D. Mais ce n'est pas le cas pour leur Anti-composant, qui bien qu'ayant une spécification de composant sont en fait constitués de plusieurs objets. Et cette composition peut être riche et variée, bien plus riche et variée que la trace à l'interface ne peut le laisser supposer.

Il nous faut donc ici faire une distinction. Toutes les traces aux interfaces ont des structures de spécifications de composants (en fait de couples composant/Anti-composant), mais seules quelques rares interfaces enveloppent un objet que nous pourrons appeler "**primitive**", pour signifier qu'il s'agit d'un objet élémentaire de l'implémentation. Remarquons que ce n'est pas la forme de la spécification qui est déterminante puisque par exemple ici la trace $[b'?; b!]^\infty$ n'est pas la trace d'une primitive. En effet, de part et d'autre de cette interface se trouvent deux primitives.

Pour ce qui est des *primitives* (ici les composants A, B, C et D) nous pouvons remarquer deux propriétés :

1. Toutes les projections de la trace complète (ici $[a; c; b'; (d \parallel e); b; c]^\infty$) sur toutes les interfaces de primitives doivent être compatibles avec les spécifications de chaque primitive. Ainsi, la projection sur l'interface-frontière de chaque primitive doit être la même spécification que lorsqu'il est en couple avec son "Anti-" (chaque composant primitif doit croire qu'il a affaire à son "Anti-").
2. Une et une seule primitive est "Inverted-" dans l'implémentation d'une phase, c'est celle pour laquelle le premier symbole du crochet de la phase complète correspond à une sortie. Lorsque le montage est la synchronisation de plusieurs phases, alors il peut y avoir un composant Inverted par phase (voir par exemple la figure 20).

Remarque : Cette notion de composition/décomposition, qui a la particularité d'être *définie à partir de phases* simples (c'est-à-dire de traces répétitives infinies), et non à partir de traces quelconques, a la propriété de

déboucher sur des montages "Live & Safe", grâce aux propriétés précédentes. Parmi les deux conditions mentionnées ci-dessus pour les primitives, la première propriété assure ce qu'Ebergen appelle "non-interférence de calcul" [7]. La seconde propriété est la condition d'initialisation de Linder/Harden qui impose un et un seul jeton par boucle [26].

3.3 Phases dissymétriques obtenues par composition

La notion de "composants" que nous venons de dégager est issue d'une "décomposition d'une phase", suggérée par l'écriture des traces. Par exemple, le symbole "||" dans l'écriture d'une trace induit une implémentation à l'aide d'un assemblage "C-Muller + Fork", et cela induit qu'il pourrait exister ces composants, "C-Muller" d'une part et "Fork" d'autre part. Mais nous n'avons jusqu'à maintenant pas encore considéré que nos "composants primitifs" pouvaient être pris isolément, indépendamment de leur "Anti". Il est maintenant bien tentant de définir une **composition**, par assemblage de composants primitifs orientés déjà vus, sans pour autant expliciter constamment les "Anti".

Il n'est pas nécessaire de se limiter aux composants "orientés". Il sera intéressant de garder à l'esprit que les compositions ainsi construites pourraient éventuellement correspondre à des traces symétriques droite-gauche et donc pourraient permettre des implémentations non-orientées.

Quels nouveaux calculs allons-nous construire par ces compositions ? Allons-nous nous restreindre à ce que construisent les phases ? Comment assurerons-nous que les montages soient "Live & Safe" ?

Nous savons par ailleurs qu'un composant électronique ne peut jamais être spécifié sans que cela entraîne des contraintes sur son environnement. En effet, les signaux d'entrée d'un composant doivent respecter des calibrages en tension, des durées minimales, ne doivent pas changer en même temps qu'un autre signal, etc. Ceci n'est pas réservé au monde des objets physiques, puique nous savons qu'en mathématiques, la définition d'une fonction entraîne des contraintes sur son "domaine de définition". Nous avons vu également que la composition de Ebergen doit respecter une règle de "non-interférence de calcul" qui restreint les compositions dans lesquelles un composant peut être utilisé.

Sans le justifier davantage maintenant, nous choisirons de nous fixer les contraintes suivantes pour nos compositions :

- Chaque composition devra construire une phase, ou plusieurs phases synchronisées. Nous n'envisagerons donc que des compositions rebouclées qui répéterons des actions indéfiniment.
- Les projections de la trace complète de la phase (ou des phases) sur toutes les interfaces possibles seront compatibles avec les spécifications

de tous les composants.

– La présence de composants "Inverted" sera compatible avec la condition "Live & Safe" déjà discutée. On peut dire comme Linder/Harden : "un et un seul jeton par boucle". Ceci conduit à : "un et au plus un composant 'Inverted' par phase". Nous écrirons la trace complète de la phase en commençant par le symbole qui correspond à la sortie de ce composant (il peut s'agir de plusieurs symboles, par exemple si c'est un Inverted-Fork).

Afin de construire de telles compositions quelconques, à partir des phases élémentaires déjà vues, nous allons montrer que nous pouvons n'utiliser que des **insertions de Wire** ou des **substitutions**, deux opérations que nous allons définir maintenant.

Ceci va nous conduire à définir de nouveaux "composants" comme assemblages de sous-parties de ces compositions. Les nouveaux "composants" ainsi définis seront davantage contraints que ceux définis par une composition classique, comme celle de Ebergen (cette dernière essayant une métaphore avec l'atomisme). Davantage contraints car nous nous imposons de toujours partir de phases, c'est-à-dire de boucles fermées, entièrement connues, construites par assemblages de ces boucles. Pour nous, nous passerons donc toujours par l'étape intermédiaire qui consiste à utiliser des composants connus pour construire une nouvelle phase (ou plusieurs), puis à définir le nouveau composant par décomposition de ces nouvelles phases. D'où la règle suivante :

Règle : Un composant ne pourra être défini que par décomposition de phases.

3.4 Dissymétrie par insertion d'un Wire

Considérons la phase (non-dissipative) :

$$[a; b]^\infty$$

implémentée par les composants orientés de la figure 7, le Wire $[a?; b!]^\infty$ et l'Inverted-Wire $[a!; b?]^\infty$.

Insérons le symbole c entre a et b. Ou bien on peut dire : remplaçons le symbole b par "$c; b$". Remarquons que ceci rend la phase dissipative.

$$[a; c; b]^\infty$$

Cette phase peut être implémentée comme sur la figure 18, qui satisfait les contraintes de composition que nous avons imposées. Ceci définit 3 interfaces, c'est-à-dire 3 découpages possibles en composants, ou 3 projections : $[a; b]^\infty$, $[c; b]^\infty$, $[a; c]^\infty$.

Comme chacune de ces projections encadre une primitive, nous pouvons en hériter les orientations des bornes et cela nous donne les spécifications des

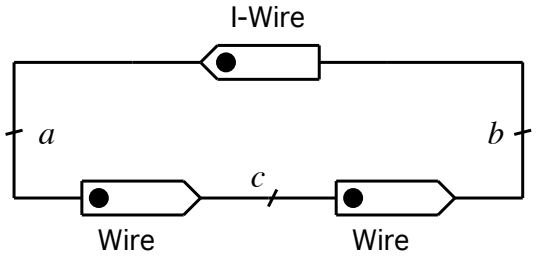
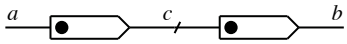


FIGURE 18 – Insertion d'un Wire dans une phase

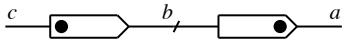
3 composants : $[a!; b?]^\infty$ (Inverted-Wire), $[c?; b!]^\infty$ (Wire) et $[a?; c!]^\infty$ (Wire). Nous constatons qu'il n'y a bien qu'un seul composant "Inverted" dans la boucle.

Ces 3 projections sont symétriques droite-gauche donc non-dissipatives. Nous pouvons en déduire : L'insertion d'un Wire dans une phase ajoute de la dissipation (un symbole par cycle).

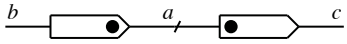
Nous pouvons aussi considérer de nouveaux composants, obtenus maintenant par décomposition de cette nouvelle phase. Cela nous introduit à ce que peut être l'association de composants primitifs :



$[a?; c; b!]^\infty$ vu comme le Wire $[a?; b!]^\infty$



$[a!; c?; b]^\infty$ vu comme l'I-Wire $[a!; c?]^\infty$



$[a; c!; b?]^\infty$ vu comme l'I-Wire $[c!; b?]^\infty$

Une association de composants qui comprend un composant "Inverted" se comportera comme un Inverted.

L'"Insertion d'un Wire" peut être aisément généralisée en considérant que tout symbole permet l'insertion d'un Wire. Par exemple, le symbole $...a...$ est remplacé par $...a; a'...$ (figure 19).

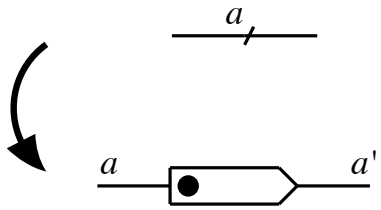


FIGURE 19 – Insertion d'un Wire sur un symbole

Nous pouvons remarquer que cette insertion conserve les propriétés qui

conduisent à assurer que cette composition est bien "Live & Safe".

En revanche, on ne peut insérer un Inverted-Wire car cela devrait correspondre à un nouveau symbole qui prétendrait pouvoir prendre la position au début du crochet, qui est déjà occupée dans toute phase. Ainsi, l'association de deux Inverted-Wire en série dans une boucle est interdite.

3.5 Déplacement du caractère "Inverted" par rotation dans la phase

Dans le crochet qui écrit la phase, on peut effectuer une rotation des symboles (ou des groupes de symboles) séparés par des ";". Cela ne change pas la phase (car ne change pas la trace infinie), mais cela correspond à une implémentation différente, dans laquelle le caractère "Inverted" est décalé d'un composant vers un autre. Par exemple, dans la figure 20, on a représenté deux implémentations de la phase $[a; (b \parallel c)]^\infty$. La première correspond à $([c; a; d]^\infty) \parallel ([b; a; e]^\infty)$, qu'on peut écrire aussi $[(b \parallel c); a; (d \parallel e)]^\infty$. La deuxième correspond à $[a; (b \parallel c); f]^\infty$.

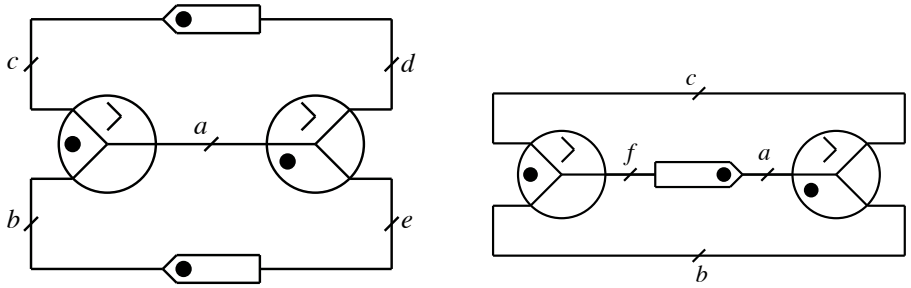


FIGURE 20 – La phase $[a; (b \parallel c)]^\infty$ avec des Inverted-Wires

Dans ces implémentations, le caractère "Inverted" est porté par des Inverted-Wires. Les deux autres composants sont donc un C-Muller et un Fork. On aurait pu également illustrer notre propos avec des montages sans Inverted-Wire, en utilisant un Inverted-C-Muller ou bien un Inverted-Fork. Par exemple la phase écrite :

$[a; (b \parallel c)]^\infty$

correspondrait à un Inverted-C-Muller associé à un Fork. Toute rotation de l'écriture de la phase correspond à un déplacement du caractère Inverted sur un composant différent.

Mais d'après ce que nous avons vu précédemment, nous pouvons toujours insérer un ou plusieurs Wire(s) dans une phase. Nous pouvons ensuite sélectionner un Wire dans chaque boucle, afin de lui transférer le caractère "Inverted", par rotation dans cette boucle. Ainsi, on peut toujours s'arranger pour que le caractère "Inverted" ne soit porté que par des Inverted-Wires.

Cette remarque peut nous permettre éventuellement d'alléger notre catalogue de composants primitifs, en ne conservant que le Inverted-Wire comme composant Inverted. (Il faut alors penser à l'impact sur la dissipation.)

Remarquons que si nous faisons ce choix, c'est-à-dire si seuls les Inverted-Wires sont Inverted, alors la condition qui assure le caractère "Live & Safe" revient à dire que dans chaque boucle doit se trouver un et un seul Inverted-Wire. Nous nous retrouvons alors dans une situation identique à celle des schémas de Ebergen [7]. Dans ce cas, pour la lecture d'un schéma, dans chaque boucle c'est le Inverted-Wire qui nous indique le sens de rotation. Il n'est alors plus nécessaire de noter le sens de propagation sur les autres composants.

3.6 Substitution sur interfaces identiques

Supposons que nous connaissions déjà (figure 21) deux compositions (orientées) distinctes C_1 et C_2 comportant chacune une interface (respectivement I_1 qui découpe C_1 en A_1, B_1 et I_2 qui découpe C_2 en A_2, B_2), telles que les traces respectives T_1 et T_2 à ces interfaces sont identiques modulo un renommage.

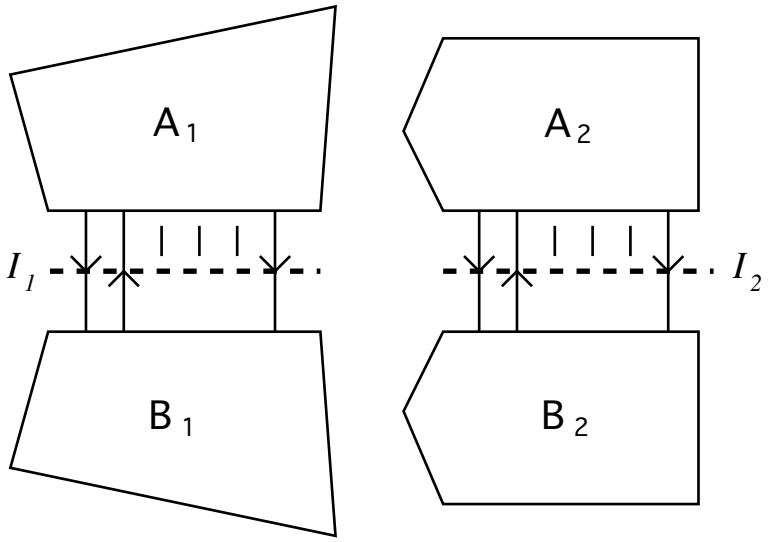


FIGURE 21 – Deux compositions (C_1 à gauche et C_2 à droite)

Supposons en outre que pour les composants A_1, B_1 d'une part et A_2, B_2 d'autre part, les bornes d'entrées et sorties se correspondent modulo ce renommage. (Remarque : si A_i est l'Anti-composant de B_i pour ce qui est de la trace, il peuvent avoir des structures internes très différentes de celles qui correspondraient par symétrie, c'est-à-dire que les symboles cachés sont en général différents.)

Alors, nous nous permettrons de considérer les deux nouvelles compositions : A_1 avec B_2 et A_2 avec B_1 (figure 22), les entrées/sorties étant associées suivant la bijection obtenue après renommage éventuel.

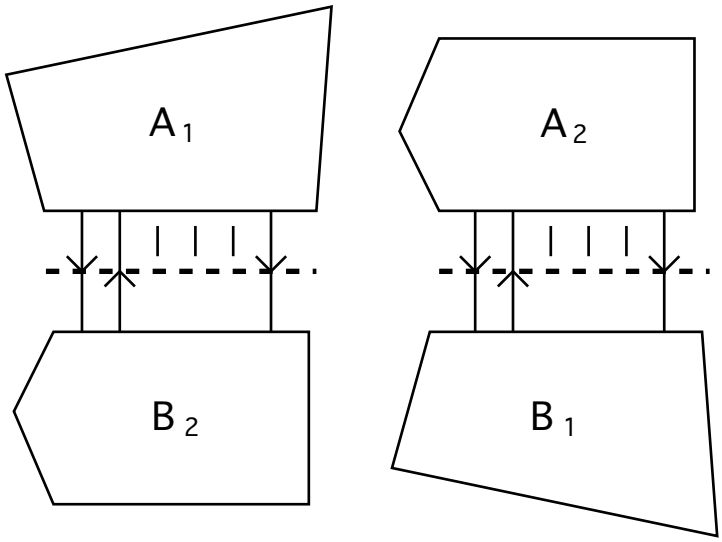


FIGURE 22 – Deux nouvelles compositions

Un exemple d'une telle substitution

Considérons les deux compositions de la figure 23 qui ont des interfaces identiques sur les lignes en pointillé. Ces deux interfaces définissent des couples Wire/Inverted-Wire. (Notons que celle de droite comporte en fait deux boucles synchronisées) :

Considérons leurs traces complètes, et leurs projections sur les interfaces en pointillés :

	Trace complète	Projection sur les interfaces
C_1	$[c; a; b; c; d]^\infty$	$[a; b]^\infty$
C_2	$([a'; e; b']^\infty) \parallel ([g; e; f]^\infty)$	$[a'; b']^\infty$

Comme les traces aux interfaces et les entrées/sorties se correspondent par renommage de a' en a et b' en b, alors nous pouvons associer A_1 et B_2, et nous obtenons la composition de la figure 24, dont la trace complète est :

$$([c; a; e; b; c; d]^\infty) \parallel ([g; e; f]^\infty)$$

Cette dernière trace, projetée sur l'alphabet de A_1 donne la trace de A_1, et projetée sur l'alphabet de B_2 donne la trace de B_2 (en identifiant a et a', et en identifiant b et b').

Cette substitution permet donc de créer des montages arbitrairement complexes à partir de montages plus simples en conservant la propriété "Live & Safe" par construction.

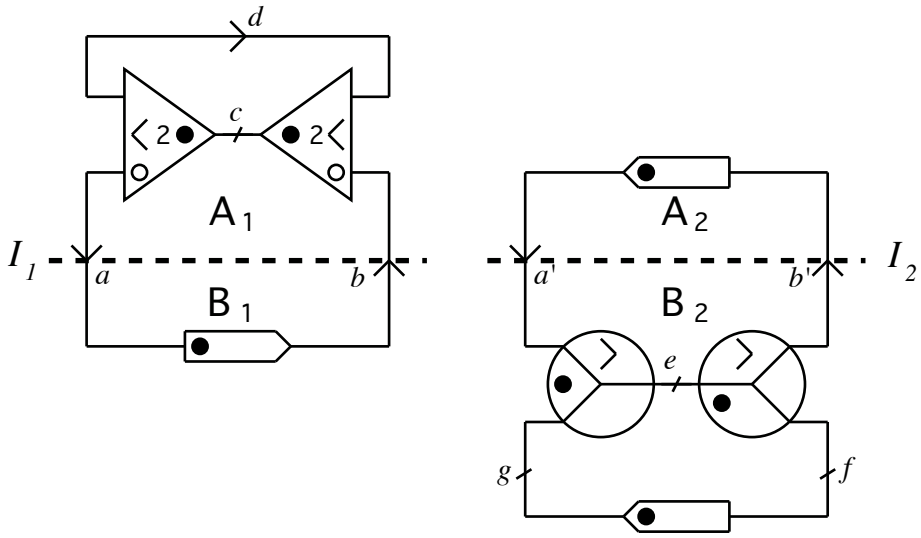


FIGURE 23 – Deux compositions : C_1 à gauche et C_2 à droite

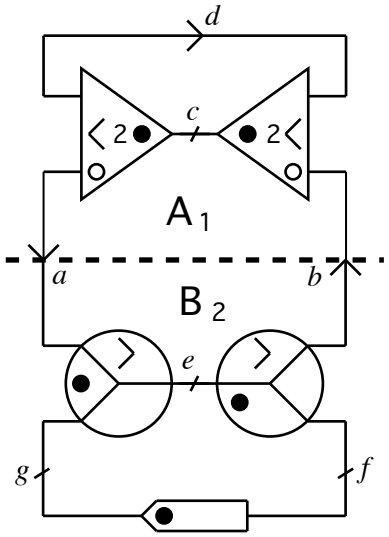


FIGURE 24 – Une nouvelle composition

3.7 Substitution sur interfaces compatibles : notion d'*implémentation*

Reprenons le schéma de la figure 21, et les mêmes notations, mais cette fois-ci, les traces T_1 et T_2 ne peuvent pas s'identifier par renommage. Toutefois, il existe entre T_1 et T_2 une relation qui possède les propriétés suivantes (nous supposerons que dans ces compositions, ce sont A_1 et A_2 qui sont "Inverted", c'est-à-dire que le crochet de la phase commence par un symbole qui est pour eux une sortie) :

- Il existe un renommage qui associe bijectivement chaque entrée de A_1 à une entrée de A_2 et chaque sortie de A_1 à une sortie de A_2.
- Modulo ce renommage, toute sortie de A_1 est acceptée par B_2 et déclenchera comme sortie de B_2 un symbole accepté par A_1. ("Accepté" signifie : compatible avec la spec.)
- Cette propriété n'existe pas entre A_2 et B_1.

Nous dirons alors que B_2 **implémente** B_1, et nous autoriserons l'association entre A_1 et B_2 pour créer une nouvelle composition. (Alors que l'association A_2 avec B_1 est impossible.)

Exemple : Le Merge implémente l'Elggot

Considérons la figure 25 qui implémente la trace $[a; c; b; c]^\infty$. L'Inverted-Toggle joue le rôle de A_1, et le Merge joue le rôle de B_2. Un Elggot jouerait le rôle de B_1, et un Inverted-Décideur jouerait le rôle de A_2.

T_1 : $[a; c; b; c]^\infty$
T_2 : $[(a|b); c]^\infty$

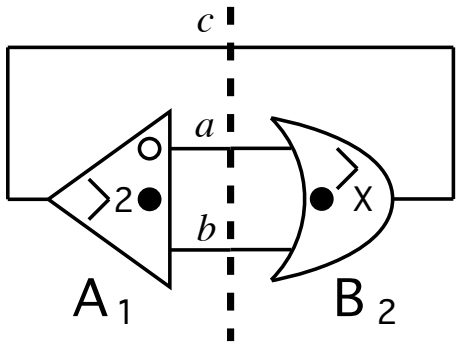


FIGURE 25 – Implémentation d'un Elggot par un Merge

Cette implémentation est dissipative, car le Merge est dissipatif. Alors que le couple Inverted-Toggle/Elggot n'est pas dissipatif. Nous conjecturerons la généralisation en notant que l'usage de cette "implémentation" peut avoir pour effet d'augmenter la dissipation physique.

Quand nous disons cela, nous supposons que le composant Merge est intrinsèquement dissipatif. Alors que sa trace dans ce montage particulier

est symétrique droite-gauche. Est-ce correct ? Nous considérons que le composant Merge oublie à chaque fois le symbole d'entrée précédent, alors que l'Elggot n'oublierait pas. Nous avons ici en tête une référence à [27, 28, 29], qui associent dissipation à oubli d'information.

3.8 Associativité des composants

Considérons les deux montages de la figure 26. Ce sont deux compositions qui ont même trace ($[a; (b|c|d)]^\infty$) sur l'interface en pointillé (car l'opérateur "|" est associatif) et nous pouvons donc appliquer la "substitution sur interfaces identiques".

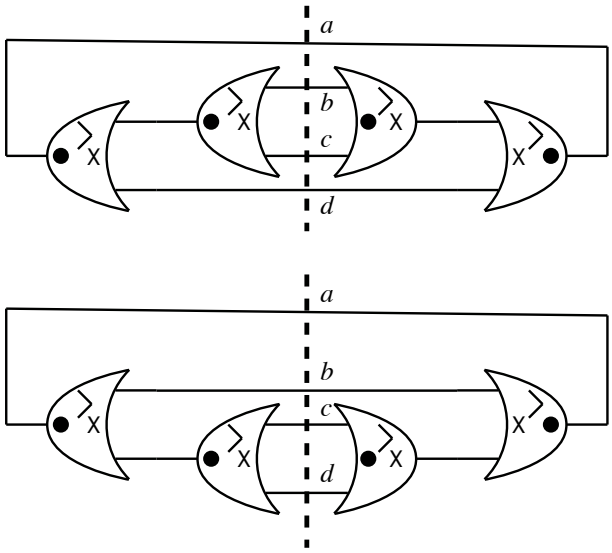


FIGURE 26 – Deux implémentations de $[a; (b|c|d)]^\infty$

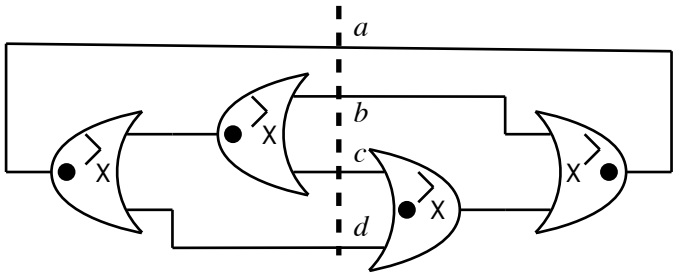


FIGURE 27 – Une autre implémentation de $[a; (b|c|d)]^\infty$

Nous obtenons alors par exemple le montage de la figure 27. Cela revient à dire qu'*on peut permuter deux Décideurs successifs ou bien qu'on peut*

permuter deux Merges successifs. On peut dire aussi qu'on peut envisager des Décideurs à 3 sorties (ou davantage) ainsi que des Merges à 3 entrées (ou davantage). Nous avons ainsi transféré l'associativité des opérateurs sur les traces vers l'associativité des composants.

3.9 Implémentation des N-Div par substitution

Considérons la composition de la figure 28, qui implémente un Wire $[a?; b!]^\infty$ à l'aide d'un (N-1)-Div et d'un Merge.

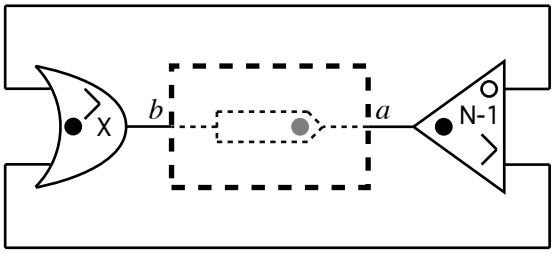


FIGURE 28 – Un Wire $[a?; b!]^\infty$

Considérons par ailleurs la composition de la figure 29, qui implémente un Inverted-Wire $[a!; b?]^\infty$ à l'aide d'un Inverted-Toggle et d'un Merge.

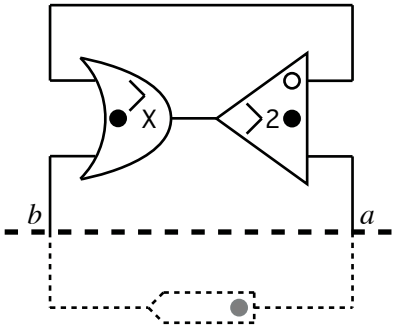


FIGURE 29 – Un Inverted-Wire $[a!; b?]^\infty$

Associons ce Wire et cet Inverted-Wire par une "substitution sur interfaces identiques", nous obtenons le schéma de la figure 30.

Nous pouvons maintenant utiliser l'associativité pour répartir autrement les entrées des deux Merges afin d'obtenir la composition de la figure 31.

Changeons un peu le dessin, sans changer le montage : permutons les sorties du (N-1)-Div et faisons apparaître une interface (en pointillés) pour obtenir la figure 32, dans laquelle nous avons renommé les bornes des composants d'une façon nouvelle.

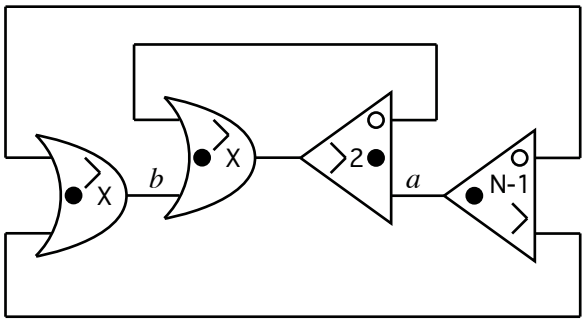


FIGURE 30 – Une implémentation de la phase $[a; b]^\infty$

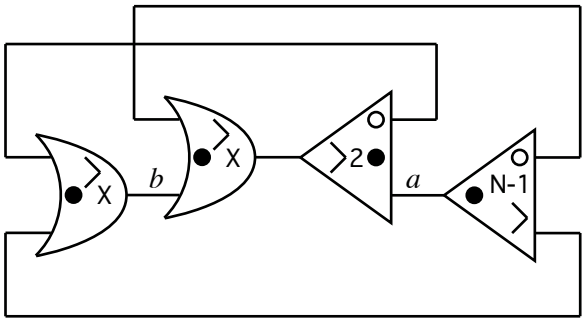


FIGURE 31 – Une nouvelle composition

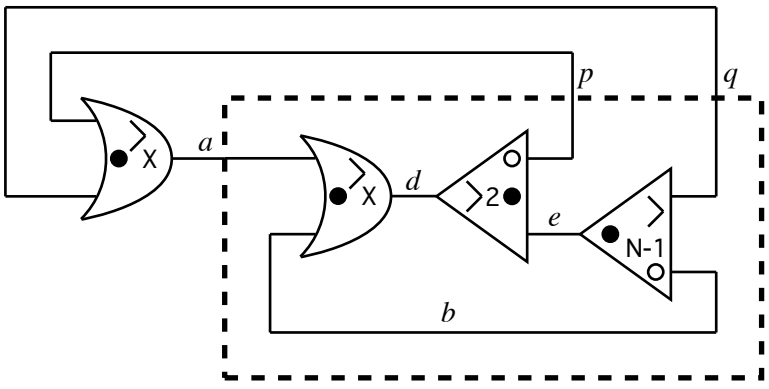


FIGURE 32 – Une nouvelle interface

La trace complète de la phase de la figure 32 est, en utilisant la définition du (N-1)-Div :

$$\left[p; a; d; e; [b; d; p; a; d; e]^{N-2}; q; a; d\right]^{\infty}$$

Si nous projetons cette trace sur l'interface en pointillés, nous obtenons :

$$\left[p; a; [p; a]^{N-2}; q; a\right]^{\infty}$$

qui est la définition du N-Div :

$$\left[[p; a]^{N-1}; q; a\right]^{\infty}$$

.

Nous venons donc de démontrer qu'on peut implémenter un N-Div à l'aide d'un (N-1)-Div, d'un Toggle et d'un Merge.

Par récurrence, nous pouvons donc implémenter tous les N-Div à l'aide de Toggles et de Merges. C'est une généralisation du "diviseur par 3" de Ebergen [7]. Remarquons que nous avons obtenu ce résultat par un raisonnement sur des phases, c'est-à-dire des boucles complètes, par un raisonnement qui conserve la propriété "Live & Safe" à toutes les étapes de la démonstration. Alors que les auteurs précédents réalisent une composition qui ne considère pas la partie "Anti" (que l'on peut appeler environnement) et doivent à la fin contrôler l'aspect "Live & Safe", en les liant à des contraintes sur l'environnement.

Nous prétendons ainsi avoir défini une méthode de conception de montages qui sont "Live & Safe" par construction.

Remarque : Les implémentations de l'exemple précédent sont dissipatives car elles comprennent des Merges, alors que les N-Div primitifs ne le sont pas puisque leurs traces sont symétriques droite-gauche.

On pourrait se demander s'il ne serait pas possible de trouver une autre implémentation des N-Div à partir de Toggles mais sans Merge, afin d'essayer de trouver une implémentation non-dissipative (mais toutefois par récurrence) pour ces objets qui ne sont pas dissipatifs dans leur définition. Or, si nous analysons notre démonstration, nous pouvons remarquer qu'elle utilise l'*implémentation* d'un Elggot par un Merge, et qu'ensuite cela permet d'utiliser l'*associativité* des Merges. Si on utilisait des Elggot à la place des Merges, on ne pourrait pas effectuer la permutation permise par l'associativité car les Elggots ne sont pas associatifs. Ceci ne nous donne donc pas de piste pour trouver une implémentation non-dissipative (ce qui ne nous prouve pas qu'il n'y en ait pas). Pouvons nous conjecturer qu'il n'y en a pas ?

Remarque : Pouvons nous nous autoriser à dire que dans la figure 32, c'est l'"intérieur" du pointillé qui réalise la division par N, plutôt que l'extérieur, ou tout l'ensemble ? Pour que ce ne soit que l'intérieur, peut-être faudrait-il que l'interface soit un axe de symétrie du schéma (c'est-à-dire que

l'extérieur soit symétrique de l'intérieur) ? Essayons d'associer l'intérieur du pointillé à son image miroir (par rapport à l'interface considérée). Cela ne fonctionnera pas car le Merge intérieur sera transformé en Décideur extérieur et les séquences ne seront pas forcément respectées.

Faut-il nous résoudre à considérer que *c'est tout l'ensemble de la figure 32 et pas seulement une partie qui implémente cette fonction N-Div* ? Nous sommes tentés de le dire.

Ou bien sinon, on peut dire que c'est uniquement l'intérieur, mais à condition que l'environnement respecte la contrainte d'être équivalent à un Merge. Il y a donc deux possibilités : ou bien on dit que la "fonction" d'un composant (ce qu'il "fait"), définie par sa spécification, n'est pas propre à ce qui se trouve d'un seul côté de l'interface (sur laquelle est définie la spécification) ; ou bien on *cache la participation active de l'extérieur* sous la forme d'une "contrainte sur l'environnement". En fait, la fonction est celle du couple composant-environnement, si on appelle "environnement" ce qui est de l'autre côté de l'interface qui sert à définir le composant. De ce point de vue, les interfaces qui séparent une phase en deux composants image-miroir l'un de l'autre sont particulières : il suffit de regarder un seul côté pour connaître la fonction. (Elles n'ont pas besoin d'être orientées et elles ne sont pas dissipatives.)

3.10 Sequencer de Ebergen

À partir de phases simples déjà rencontrées, nous pouvons aisément construire, par substitutions, la composition de la figure 33.

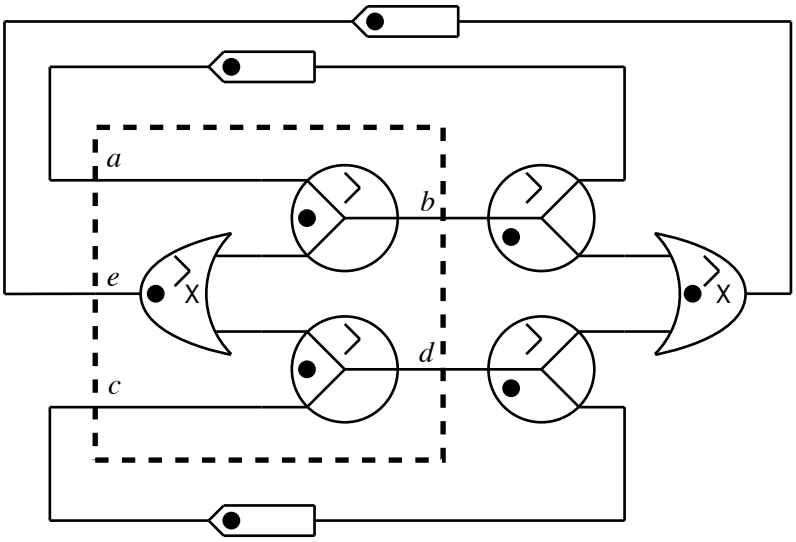


FIGURE 33 – Sequencer

La trace à l'interface I est :

$$([a;b]^\infty) \parallel ([c;d]^\infty) \parallel ([e;(b|d)]^\infty)$$

Nous reconnaissons la spécification du "Sequencer" de Ebergen [7]. L'intérieur du pointillé peut être considéré comme une implémentation du composant Sequencer, car la composition que nous avons dessinée à l'extérieur lui est symétrique (modulo les Inverted-Wires qui ne font que synchroniser les 3 phases).

Or, il se trouve que cet extérieur est précisément le même schéma que celui de l'implémentation que Ebergen donne de son "Token-Ring" [7]. Mais pouvait-il en être autrement ? Puisque le séquencer ne peut avoir pour environnement qu'un montage qui se comporte comme ça.

3.11 Engendrer tous les calculs par composition de primitives ?

Considérons la liste suivante de composants orientés, établie dans un souci de minimisation du nombre de primitives :

- Wire et Inverted-Wire
- C-Muller et Fork
- Toggle
- Décideur et Merge

Dans cette liste, le seul "Inverted" est le Inverted-Wire, les seuls qui sont "intrinsèquement dissipatifs" sont le Décideur et le Merge.

Nous avons vu d'autres composants qu'on peut engendrer ou implémenter à partir de la liste ci-dessus par composition puis décomposition. Nous ne les avons pas fait figurer dans cette liste, puisque nous sommes ici dans un souci de minimisation du nombre de primitives. Il y aurait :

- Les autres "Inverted" ne sont pas indispensables car on peut reporter le caractère "Inverted" sur des Inverted-Wires, par rotation de l'écriture de la trace
- Les N-Divs (pour $N > 2$) peuvent être implémentés à l'aide de Toggles et Merges,
- L'Elggot peut être implémenté à l'aide du Merge,
- Le Sequencer de Ebergen peut être implémenté à l'aide de Décideur, C-Mullers et Forks.

Notons que *ces 4 causes de raccourcissement de la liste de primitives sont toutes des causes d'augmentation de la dissipation.* Il ne sera donc pas toujours judicieux de choisir une liste courte de primitives lors des implémentations physiques.

Notons que notre liste diffère de celle de Ebergen uniquement par le choix du Décideur à la place du Sequencer. Notre choix est justifié par la mise en évidence de la symétrie entre Décideur et Merge. En tous cas, si la

liste de Ebergen suffit à engendrer tous les calculs asynchrones, alors notre liste suffit également.

Conjecture : Avec ce jeu de composants, dits "*primitifs*", nous pouvons engendrer toutes les phases permises par toutes les combinaisons des symboles et des opérateurs de base, en utilisant pour former de nouveaux composants uniquement l'*insertion de Wire* et la *substitution* que nous avons définies.

Parmi les phases engendrées, certaines ont la propriété de connaître une interface telle que de part et d'autre les composants sont image-miroir l'un de l'autre (l'un est "Anti-" de l'autre). Ces phases ont des traces symétriques droite-gauche et peuvent éventuellement connaître une implémentation non-orientée non-dissipative.

Calcul : ce que nous avons vu montre qu'il ne faut pas considérer l'implémentation d'un calcul comme un assemblage (ou composition) d'objets simples indépendants qui aboutit à un objet complexe (le calcul complet), comme dans la métaphore du Lego ou de l'atomisme. Mais il faut voir au contraire une *décomposition*.

Cette façon de présenter les choses semble indiquer qu'on n'envisagerait que des calculs qui sont des boucles infinies. Or il suffit de considérer qu'un calcul fini peut être répété pour que les calculs finis soient traîtés dans le même modèle (figure 34). Un calcul serait alors équivalent à un Wire.

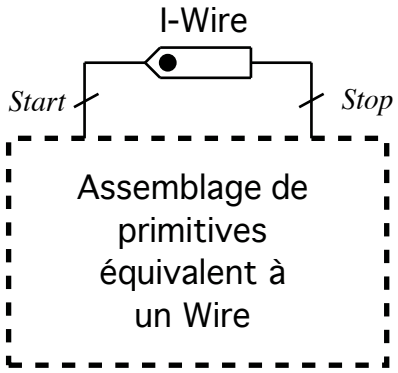


FIGURE 34 – Un calcul fini

En fait, il y a deux extrémités qui sont simples : d'une part celle qui est constituée de la (ou les) phase globale (la plus externe qui généralement n'est qu'une boucle Start/Stop qui est réitérée à l'infini), d'autre part celle qui est constituée des objets primitifs élémentaires. Entre ces deux extrêmes simples, tout un réseau d'interfaces qui peut être extrêmement complexe. À chaque nouvelle action de décomposition, on ajoute des variables, ce qui augmente le nombre d'interfaces de façon exponentielle. La trace à une interface quelconque de cet ensemble n'a de sens que pour le calcul global. La

notion de temps qui émerge est celle liée à la dissipation et son lien avec la progression dans les traces des interfaces dissipatives.

Dans ces objets, à tous les niveaux, nous vérifions :

- La *non-interférence de calcul* de Ebergen (sur toutes les **primitives**) [7].
- Les *règles de Udding* (sur toutes les **interfaces**) [5].
- Les conditions *Live & Safe* de Linder/Harden (sur toutes les **phases**) [26].

Ces 3 types de conditions sont vérifiées automatiquement par construction si le calcul est défini par la décomposition que nous avons décrite, à partir d'une phase simple.

Reste à construire les topologies rendues possibles par ces espaces.

4 Exemples de topologies

4.1 Substitutions de quadripôles : parallélisme, synchronisation et propagation

Considérons les deux phases suivantes (figure 35), indépendantes :

$$([a;b]^\infty) \parallel ([c;d]^\infty)$$

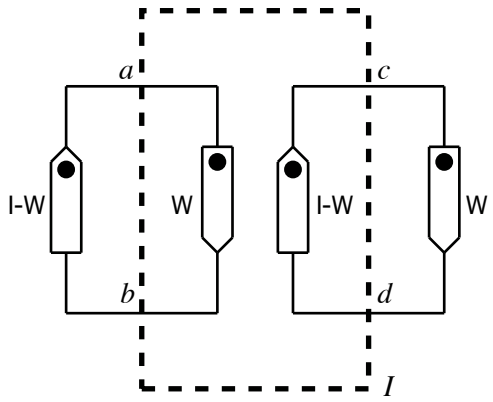


FIGURE 35 – Deux phases indépendantes

Les deux phases n'ont pas de symbole en commun. Nous dirons qu'il y a **parallélisme** (ou concurrence), mais **sans synchronisation** entre les deux phases.

Transformons par une substitution l'"intérieur" de l'interface I de la figure 35 comme indiqué sur la figure 36. Nous obtenons un schéma qui n'est pas équivalent, mais qui est une *implémentation*, et dont l'écriture est :

$$[a;c;d;b]^\infty$$

Il n'y a plus qu'une seule phase. Si nous projetons cette trace, soit sur $\{a,b\}$, soit sur $\{c,d\}$, nous obtenons séparément les traces des deux phases vues précédemment. Mais ce n'est pas pour autant que cette nouvelle phase est équivalente au schéma précédent puisque cette fois les deux phases sont entrelacées, donc synchronisées, et ne sont plus indépendantes. Il y a ici **synchronisation**, mais **sans parallélisme**. La partie centrale de la figure 36 *implémente* la partie centrale de la figure 35, mais **en ajoutant une contrainte forte** (l'entrelacement).

Remarquons que cette version de la figure 36 introduit une notion de *propagation*, de deux sortes : de la gauche vers la droite à cause du Wire $[a;c]^\infty$, et aussi de la droite vers la gauche à cause du Wire $[d;b]^\infty$.

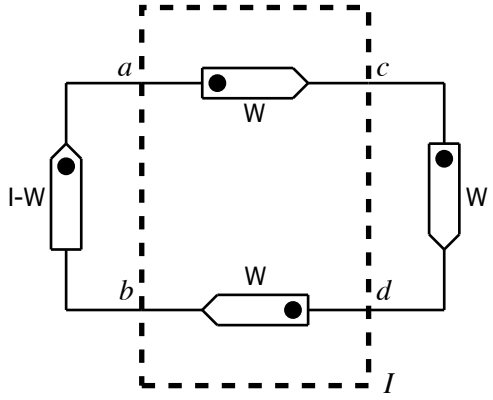


FIGURE 36 – Une phase

Substituons un autre montage à l'intérieur de l'interface I de la figure 35, en utilisant une déformation d'un schéma déjà vu (vu par exemple figure 20), pour obtenir la figure 37, qui est une autre *implémentation*, mais différente :

$$([a; f; b]^{\infty}) \parallel ([e; f; c; d]^{\infty})$$

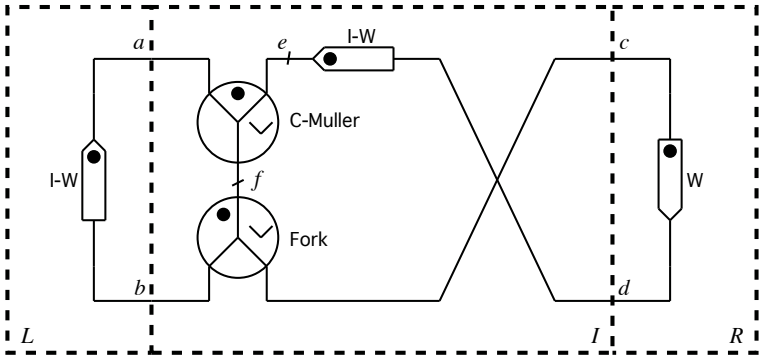


FIGURE 37 – Deux phases synchronisées

Cette fois-ci, il y a bien deux phases, donc parallélisme. Ici, les deux phases sont **synchronisées** (par l'intermédiaire du symbole commun f), mais *sans perte du parallélisme*. La partie centrale de la figure 37 *implémente aussi* la partie centrale de la figure 35, mais avec une **contrainte plus faible** que figure 36. Ce schéma est celui du **propagateur de Sutherland** [30]. Il introduit aussi des *propagations*, de a vers c (propagation des "Requests" chez Sutherland), et de d vers b (propagation des "Acknowledges" chez Sutherland).

Remarquons que ce qui, dans la figure 37, est à droite de l'interface L (l'ensemble $I + R$) est vu par l'Inverted-Wire de L comme un seul Wire

(c'est ce qu'il "voit" de son environnement). De même, ce qui est à gauche de l'interface R (l'ensemble $L + I$) est vu par R comme un seul Inverted-Wire. Ceci nous autorise, par itération du bloc substitué, à répéter le propagateur du milieu (l'intérieur de I) pour obtenir une ligne finie de n propagateurs à une dimension (figure 38). Cette ligne de n propagateurs sera dissipative, d'une façon proportionnelle à n, car il y a les symboles cachés e et f (de la figure 37).

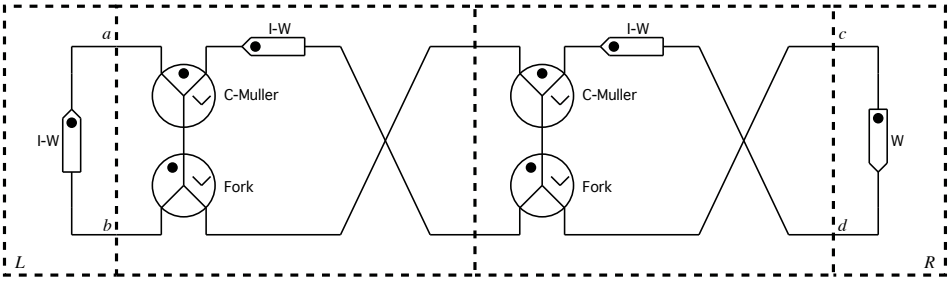


FIGURE 38 – Propagateurs dissipatifs

Mais remarquons qu'on peut réaliser une ligne infinie de propagateurs non-orientés donc non-dissipatifs si on utilise le schéma de la figure 39, qui est tiré de la répétition de la figure 9 avec substitution. Dans cette figure 39, remarquons qu'il y a symétrie droite-gauche et aussi symétrie haut-bas. Les composants du haut peuvent être des Inverted-Fork alors que ceux du bas seraient des C-Muller, mais ce peut être aussi l'inverse, ou la superposition des deux. En fait c'est leur version non-orientée.

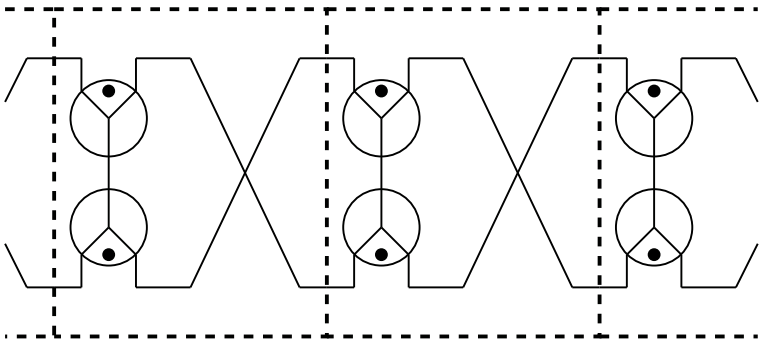


FIGURE 39 – Ligne infinie de propagateurs non-dissipatifs

Dans ce cas, si on réalise une ligne infinie de ces propagateurs (une infinité de phases), ni le sens de rotation des phases ni un **sens de propagation droite-gauche le long de la ligne horizontale** ne sont définis. Il n'y a pas de propagation, pas de dissipation, toutes les phases ont des traces

symétriques droite-gauche, donc il n'y a pas de temps qui progresse, et donc pas de "localisation de jetons dans un espace".

Par contre, s'il s'agit d'une ligne finie (figure 38), terminée à gauche par un Inverted-Wire et à droite par un Wire, alors on force un "sens de rotation" des phases et un "sens de propagation le long de la ligne de propagateurs" (ici propagation de "jetons Request" de la gauche vers la droite, et de "jetons Acknowledge" dans le sens inverse).

En fait, si nous dessinons un Inverted-Wire par propagateur (par cellule élémentaire), il y aura autant de jetons circulants que de cellules. Mais nous savons, par la réalisation de Micropipelines [30] qui utilisent ces propagateurs pour la synchronisation de Fifos élastiques, qu'on pourrait dessiner moins que un Inverted-Wire par cellule. Le nombre de Inverted-Wires définira le nombre de "places occupées" dans le Fifo. Il y aura alors une autre sorte de propagation : propagation de "places vides" de la droite vers la gauche, et propagation d'"occupants de places", de la gauche vers la droite. Ces "occupants de places" ne sont pas ce que nous avons précédemment appelé "jetons". (Pour une introduction à ces questions, voir par exemple [31].)

Dans une description comme celle de Ebergen, les Inverted-Wires sont liés à l'initialisation, c'est-à-dire à leur action au commencement du temps. Ici, il n'y a pas d'initialisation des boucles puisque les traces sont infinies à droite et à gauche, et pourtant les Inverted-Wires jouent quand même un rôle d'initialisation, mais différent. Ici dans un micropipeline, leur nombre indique le nombre de places occupées.

Notons que dans ces schémas orientés, il y a dissipation, et elle dépend du nombre de Inverted-Wires, c'est-à-dire du nombre d'"occupants de place" circulants. Alors que dans le schéma infini non-orienté, il n'y a ni propagation, ni dissipation, et nous dirons que dans ce cas il n'y a *pas de temps*.

La dissipation fait apparaître un temps qui progresse, conjointement avec une "localisation spatiale de quelque chose" qui se propage ("occupant de place"). Mais hors les instants de dissipation, il n'est pas correct de dire que quelque chose est localisé. Nous appellerons cette chaîne de propagateurs : **rayon lumineux**. Notons qu'il présente le caractère de dualité onde-corpuscule. En effet, l'aspect "corpuscule" ne se voit que en relation avec les "occupants de place", alors que le long des propagateurs, nous appellerons "onde" ce qui assure la synchronisation entre deux boucles successives (ici, ce sont plutôt les symboles qui jouent ce rôle d'ondes).

Notons l'analogie avec la physique de la propagation de la lumière : s'il n'y a pas d'absorption de photon, il n'y a pas de dissipation, mais pas non plus de propagation puisque le temps n'avance pas dans le référentiel du photon. Il faut qu'il y ait eu à la fois émission puis absorption du photon pour qu'on puisse parler de propagation.

4.2 Autres synchronisations de deux phases

Les phases de la figure 35 peuvent également être implémentées à l'aide des schémas des figures 40 et 41.

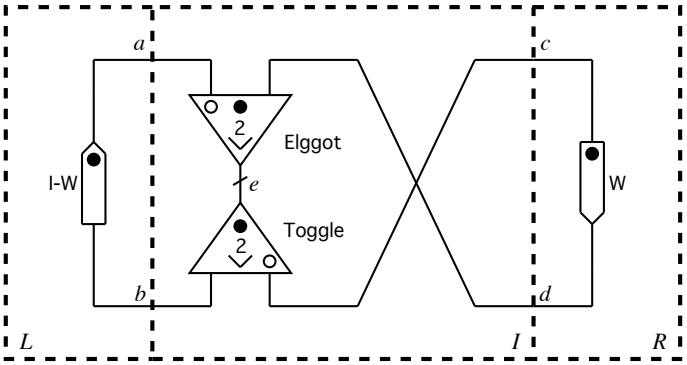


FIGURE 40 – Alternance simple

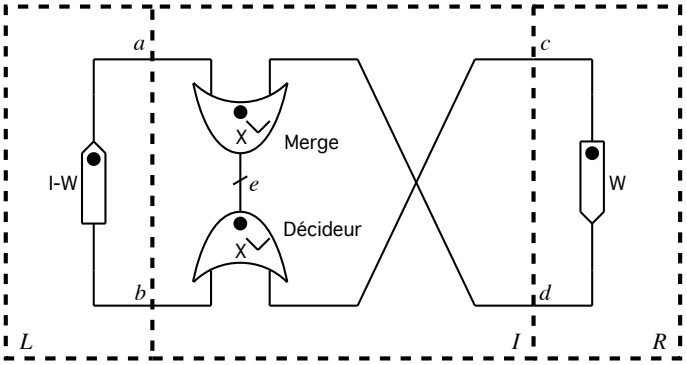


FIGURE 41 – Alternance non-régulière

Les traces complètes sont respectivement :

$$[a; e; c; d; e; b]^{\infty}$$

$$[a; e; (b^*|(c; d; e)^*)]^{\infty}$$

dont les projections sur $\{a, b, c, d\}$ sont respectivement :

$$[a; c; d; b]^{\infty}$$

$$[a; (b^*|(c; d)^*)]^{\infty}$$

Nous en déduisons qu'il n'y a pas de parallélisme car il y a une seule phase. Mais il y a synchronisation par alternance (non-régulière et dissipative dans le deuxième cas).

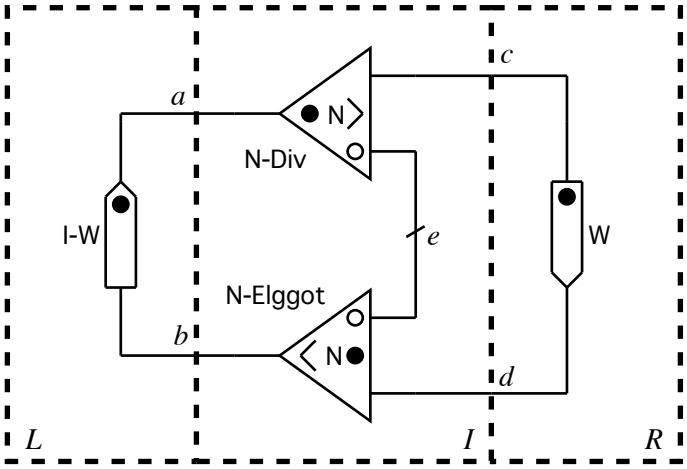


FIGURE 42 – Alternance dissymétrique

Les phases de la figure 35 peuvent aussi être implémentées à l'aide du schéma de la figure 42.

La trace complète est :

$$[a; (e; b; a)^{N-1}; c; d; b]^{\infty}$$

et sa projection sur $\{a, b, c, d\}$ est :

$$[a; (b; a)^{N-1}; c; d; b]^{\infty}$$

Nous en déduisons qu'il n'y a pas de parallélisme car il y a une seule phase. Mais il y a synchronisation par alternance, la phase de gauche étant N fois plus active que celle de droite.

Le couple N-Div/N-Elggot de cette figure peut aussi être remplacé par le couple Décideur/Merge.

4.3 Alternance dissymétrique avec parallélisme

Considérons les deux compositions classiques des figures 43 et 44 :

Les deux interfaces I_1 et I_2 ont la même trace : $[c; d; b; a]^{\infty}$ et les noms des entrées-sorties se correspondent, donc nous pouvons effectuer la substitution qui conduit à la figure 45, dans laquelle nous avons nommé I_3 l'interface commune. Dans cette figure 45, la trace complète est :

$$[e; c; ((g; h) \parallel (d; f; e; b)); a; f]^{\infty}$$

Dans cette figure 45, nous isolons une *nouvelle* interface que nous nommons I', sur laquelle la trace projetée est :

$$[e; ((g; h) \parallel (f; e)); f]^{\infty}$$

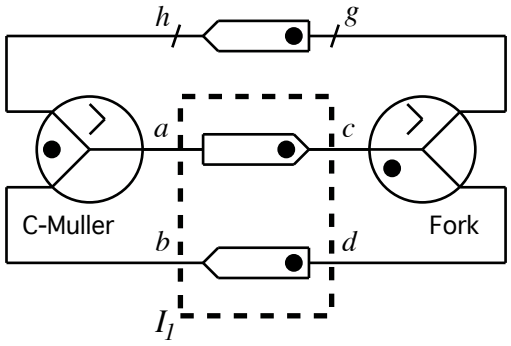


FIGURE 43 – C-Muller et Fork

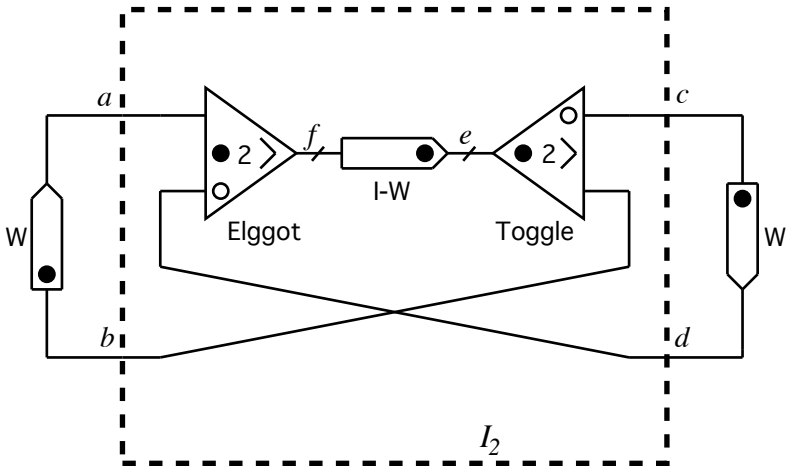


FIGURE 44 – Toggle et Elggot

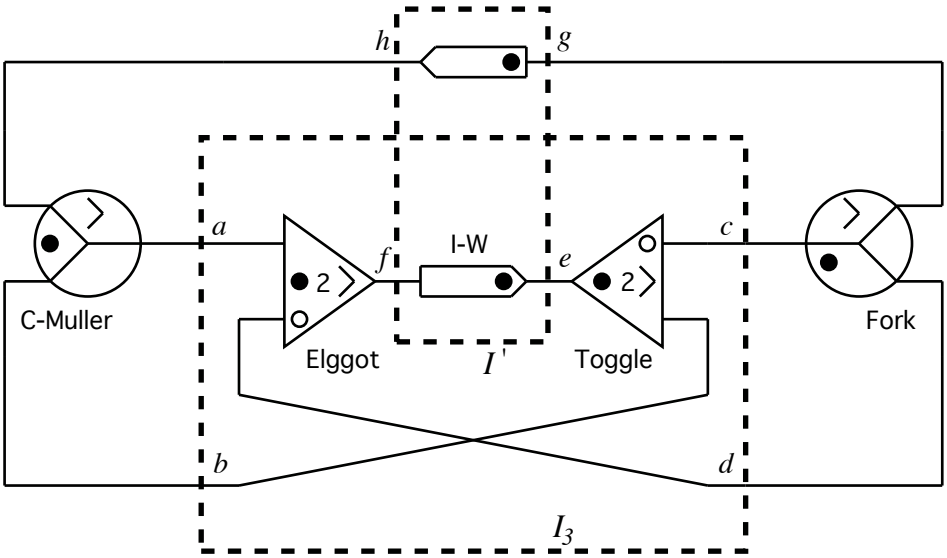


FIGURE 45 – Nouvelle composition

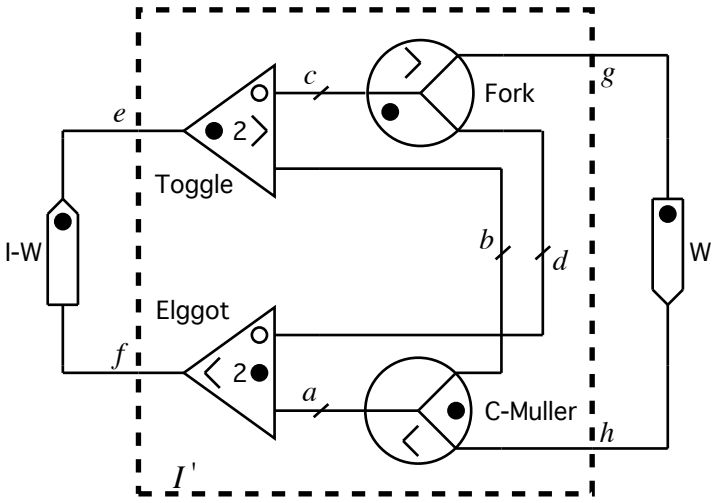


FIGURE 46 – Convertisseur dit “2-phases/4-phases”

Sans modifier ce schéma, nous le redessinons autrement (figure 46).

Nous reconnaissons alors le montage appelé traditionnellement "convertisseur 2-phases/4-phases" [32]. Dans les présentations classiques l'Elggot est habituellement implémenté par le Merge.

Dans notre terminologie, ce montage ne comprend que 2 phases. Il y a parallélisme, et c'est, comme à chaque fois, introduit par le couple "C-Muller/Fork" qui implémente deux phases synchronisées.

Notons que la substitution que nous avons définie nous fournit une bonne heuristique pour assurer que les montages fonctionnels découverts sont automatiquement "Live & Safe".

5 Discussion, conclusion

Nous avons décrit un modèle de calcul qui repose sur les traces, mais en leur faisant correspondre une signification différente de celle utilisée dans le cadre des circuits asynchrones. En effet, ici, l'objet élémentaire est la phase, et une trace décrit une combinaison de phases. Dans ce cadre, tout calcul est une boucle qui se répète indéfiniment. Ceci peut heurter nos conceptions habituelles de calcul, de temps, de composant matériel.

Les phases décrivent un ordre partiel logique qui n'est pas l'ordre temporel. Ce n'est que dans le cas particulier où l'on peut définir une propagation que la notion de temps liée à cette propagation peut ressembler à la notion de temps physique classique. Et dans ce cas, nos composants ressemblent aux composants électroniques des circuits asynchrones habituels. Dans ce cas également, la dissipation physique est bien modélisée par le nombre de symboles dans la trace (multiplié par une constante technologique comme $\frac{1}{2}CU^2$ pour les circuits CMOS par exemple). Dans ce cas également, le temps est orienté c'est-à-dire que le calcul n'est pas "renversable". Il y a donc ici un lien entre irréversibilité logique et dissipation. Ce lien est issu du langage et conduit à une structure de temps particulière.

Nous avons vu également que dans ce cadre, la dissipation est liée à la "synchronisation", si nous utilisons ce mot dans une acception particulière. Il ne s'agit pas de définir un temps global commun à tous les composants, puisque l'ordre temporel est un ordre partiel (comme dans la théorie de la relativité). Mais ici, le mot "synchronisation" fait allusion à la "coopération des boucles", c'est-à-dire qu'il décrit simplement l'ordre partiel des symboles qui résulte de cette interaction particulière entre les phases.

Le lecteur pourrait s'interroger sur cette façon de définir le calcul, qui est une répétition infinie, qui n'a pas de début ni de fin, d'opérande ni de résultat. Nous ferons remarquer que ceci est très proche de ce qui est fait en théorie des automates où le calcul est défini comme un parcours dans un graphe. Dans cette dernière théorie, si on cherche à discuter de dissipation et de réversibilité, on est également amenés à symétriser le "faire"

et le “défaire” du calcul, afin de rechercher une symétrie par renversement du temps [33, 19]. On est alors amenés à faire coïncider le début et la fin (après avoir ajouté l'image miroir du calcul), et il est alors aisé de considérer indifféremment soit une seule itération soit une réitération infinie du même calcul. Mais ce qu'apporte notre construction par rapport à celle de [33] et [19], c'est que notre temps n'est pas forcément un nombre. C'est un ordre partiel, ce qui signifie que nos composants ne sont pas repérés à l'aide d'une variable t commune à tous les composants. Nous pouvons ainsi mieux séparer ce qui est lié à une synchronisation nécessaire à un calcul particulier (l'ordre partiel particulier), de ce qui est lié à une synchronisation qui serait préalable à toute construction car découlant de la structure d'un temps pré-existant.

Il nous semble que c'est dans cette différence-là qu'on pourrait voir l'origine d'une dissipation nécessaire au calcul. Un calcul particulier crée un temps particulier, qui a sa structure propre. La dissipation est liée à ce caractère “structurel” du temps, issu de la structure de la trace qui a précédemment défini le calcul particulier.

Références

[1] A. Mazurkiewicz. “Basic Notions of Trace Theory”. In J. W. de Bakker, W. P. de Roever, and G. Rozenberg, editors, “Linear Time, Branching Time, and Partial Order in Logics and Models for Concurrency. School/Workshop, Noordwijkerhout, may-june 1988”, Number 354 in LNCS, page 285 (Springer-Verlag, The Nederlands, 1989).

[2] Jan L. A. van de Snepscheut. *Trace Theory and VLSI Design*. Number 200 in LNCS (Springer-Verlag, 1985).

[3] Charles E. Molnar, Ting-Pien Fang, and Frederik U. Rosenberger. “Synthesis of Delay-Insensitive Modules”. In “1985 Chapel Hill Conference on VLSI”, pages 67–86 (1985).

[4] Jan Tijmen Udding. *Classification and Composition of Delay-Insensitive Circuits*. Ph.D. thesis, Eindhoven University of Technology, Netherlands, September 1984.

[5] Jan Tijmen Udding. “A formal model for defining and classifying delay-insensitive circuits and systems”. *Distributed Computing*, volume 1, pages 197–204, 1986. Springer-Verlag

[6] Jo C. Ebergen. *Translating Programs into Delay-Insensitive Circuits*. Ph.D. thesis, Technische Universiteit Eindhoven, October 1987.

[7] Jo C. Ebergen. “A formal approach to designing delay-insensitive circuits”. *Distributed Computing*, volume 5, pages 107–119, 1991.

[8] Tom Verhoeff. *A Theory of Delay-Insensitive Systems*. Ph.D. thesis, Eindhoven University of Technology, 1994.

[9] Albert Einstein. "Zur Elektrodynamik bewegter Körper". *Annalen der Physik*, volume 17, pages 891–921, 1905.

[10] Albert Einstein. *The collected papers of Albert Einstein*, volume 2 (Princeton University Press, 1989), english translation edition.

[11] Bertrand Russell. *ABC of Relativity* (Routledge, 1997). Première édition : George Allen & Unwin, 1925

[12] Bertrand Russell. "Philosophical Consequences of Relativity". *Britannica*, 1926. URL `http://www.britannica.com/bps/additionalcontent/14/117887/relativity-philosophical-consequences-of`.

[13] Bertrand Russell. *The Analysis of Matter* (Routledge, 1992). Première édition : 1927

[14] Philippe Matherat and Marc-Thierry Jaekel. "Dissipation logique des implémentations d'automates - Dissipation du calcul". *Technique et Science Informatiques, http://tsi.revuesonline.com/acceuil.jsp*, volume 15(8), pages 1079–1104, octobre 1996. URL `http://hal.archives-ouvertes.fr/hal-00180863/fr/`. Traduction (en mai 1998) de cet article en langue anglaise, accessible à : `http://fr.arxiv.org/abs/quant-ph/9805018`

[15] Philippe Matherat. "Où en est-on de la dissipation du calcul ? Retour à Bennett". *Annales des télécommunications*, volume 62(5-6), pages 690–713, mai-juin 2007. URL `http://hal.archives-ouvertes.fr/hal-00082436/fr/`.

[16] Philippe Matherat. "Clockless components and relativity". *HAL - Archives ouvertes*, 29 mai 2009. URL `http://hal.archives-ouvertes.fr/hal-00389739/fr/`.

[17] Bertrand Russell. *Our Knowledge of the External World* (Routledge, 2000). Première édition : The Open Court Publishing Company, 1914

[18] Philippe Matherat and Marc-Thierry Jaekel. "Concurrent computing machines and physical space-time". *MSCS (Mathematical Structures for Computer Science), http://journals.cambridge.org/action/displayJournal?jid=MSC*, volume 13(5), pages 771–798, octobre 2003. doi :10.1017/S0960129503004067. URL `http://fr.arxiv.org/abs/cs.DC/0112020`. Ce tome est le deuxième tome d'un numéro spécial double : "The Difference between Concurrent and Sequential Computation", éditeur : Cambridge University Press

[19] Edward Fredkin and Tommaso Toffoli. "Conservative logic". *International Journal of Theoretical Physics*, volume 21(3/4), pages 219–253, 1982.

[20] Thomas J. Chaney and Charles E. Molnar. "Anomalous Behavior of Synchronizer and Arbiter Circuits". *IEEE Transactions on Computers*, pages 421–422, april 1973.

[21] L. Kleeman and A. Cantoni. "On the unavoidability of metastable behavior in digital systems". *IEEE Transactions on Computers*, volume 36(1), pages 109–112, January 1987.

[22] Leslie Lamport. "Buridan's Principle". URL `http://research.microsoft.com/en-us/um/people/lamport/pubs/buridan.pdf`.

[23] Charles L. Seitz. "Ideas About Arbiters". *Lambda*, pages 10–14, First Quarter 1980.

[24] Takayasu Sakurai. "Optimization of CMOS Arbiter and Synchronizer Circuits with Submicrometer MOSFET's". *IEEE Journal of Solid-State Circuits*, volume 23(4), pages 901–906, august 1988.

[25] David J. Kinniment. *Synchronization and Arbitration in Digital Systems* (John Wiley & Sons Ltd, 2007). ISBN 978-0470-51082-7.

[26] Daniel H. Linder and James C. Harden. "Phased Logic : Supporting the Synchronous Design Paradigm with Delay-Insensitive Circuitry". *IEEE Transactions on Computers*, volume 45(9), pages 1031–1044, september 1996.

[27] R. Landauer. "Irreversibility and heat generation in the computing process". *IBM J. Res. Develop.*, pages 183–191, July 1961.

[28] Charles H. Bennett. "The thermodynamics of computation - a review". *Int. J. of Theor. Phys.*, volume 21(12), pages 905–940, dec. 1982.

[29] Charles H. Bennett and Rolf Landauer. "The Fundamental Physical Limits of Computation". *Scientific American*, pages 38–46, July 1985.

[30] Ivan E. Sutherland. "Micropipelines". *Communications of the ACM*, volume 32(6), pages 720–738, June 1989.

[31] Jens Sparsø and Steve Furber, editors. *Principles of Asynchronous Circuit Design* (Kluwer Academic Publishers, 2001). ISBN 0-7923-7613-7.

[32] Jo. C. Ebergen, John Segers, and Igor Benko. "Parallel Program and Asynchronous Circuit Design". In G. Birtwistle and A. Davis, editors, "Asynchronous Digital Circuit Design, Workshops in Computing", pages 50–103 (BCS Springer, 1995).

[33] Charles H. Bennett. "Logical reversibility of computation". *IBM J. Res. Develop.*, pages 525–532, nov. 1973.