Les programmes prennent le temps

Marc Pouzet

Équipe Proval

LRI & INRIA & IUF

Unithé ou Café , 15 janvier 2010, Orsay

Comment programmer/contrôler des systèmes

dans lesquels le temps intervient ?

schématiquement, deux grandes familles de systèmes informatiques...

Les systèmes transformationnels

La machine attend une donnée, effectue un calcul, produit un résultat et s'arrête.

Exemples

- transformer une image couleur en image N&B
- traduire un texte
- demander un itinéraire sur une carte (ViaMichelin, GoogleMap)
- une requète Google: une entrée $\longrightarrow$ un résultat au bout d'un certain temps

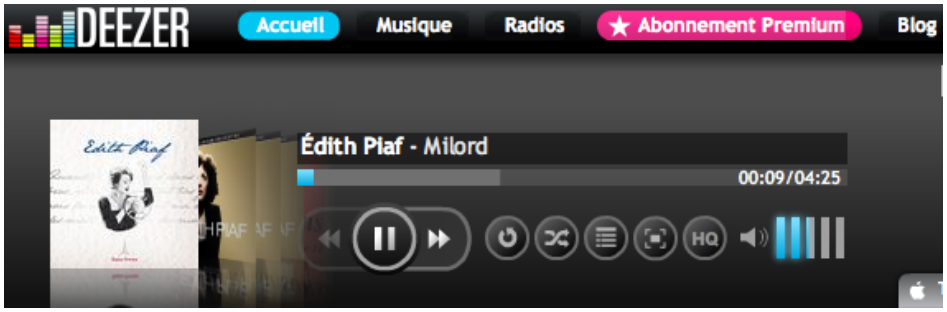
Ici, le temps ne compte pas

Les systèmes réactifs

Des interactions tout le temps; des requètes et observations en permanence

Exemples

- la gestion des fenêtres d'un ordinateur; le clavier, la souris
- un flux de musique ou de vidéo sur internet (Deezer, DailyMotion)



Si l'environnement extérieur va trop vite,

il peut attendre ou, mieux, on peut le ralentir

Ici, le temps physique ou *réel* ne compte pas vraiment

Les systèmes réactif temps réel

Contrainte supplémentaire: l'environnement ne peut pas attendre.

une voiture: allumage électronique, correction de trajectoire (ESP), système anti-patinage (ABS)

un avion: pilotage automatique par ordinateur (fly-by-wire)

Ce sont tous des programmes

Problème: commander **en même temps** l'aile gauche, l'aile droite et les ailerons arrières; contrôler roulis **et** tangage?

L'avion recoit **en même temps** les informations des centrales inertielles, des anémomètres, etc.

Il n'y a plus de transmission mécanique (cablage) entre le pilote et les gouvernes.

Comment faire avec des programmes?

Du temps et du parallélisme

- On peut lancer plusieurs fenêtres sur son ordinateur même lorsque celui-ci n'a qu'un processeur...???
- Il ne sait donc fondamentalement faire qu'une chose à la fois.
- Il donne l'illusion d'en faire plusieurs parce qu'il va très vite, beaucoup plus vite que notre propre perception au changement.
- C'est facile parce que les applications sont indépendantes les unes des autres

Est-ce que ce parallélisme nous aide pour programmer des avions et des systèmes temps réel ?

Dessinons une balançoire

```
loop
  bouger balancoire
end
```

Deux balancoires ?

Il suffit d'en exécuter deux en parallèle.

```
loop                          loop
  bouger balançoire1   ||       bouger balançoire2
end                           end
```

Pourquoi ca ne marche pas ?

Découper le temps en instants logiques

- Il décide d'exécuter un peu du premier programme puis un peu du second puis un peu du premier, etc.
- et si on ajoute une autre balançoire ?
- Le programmeur n'a pas de moyen simple de contrôler les choix faits par le système d'exploitation et que les balancoires avancent au même rythme.
- Pas de synchronisation précise dans le temps: le processeur n'en fait qu'a sa tête !

Parallélisme par *entrelacement*:

- `bouger balançoire1; bouger balançoire1;`
  `bouger balançoire1; bouger balançoire2;...`
- `bouger balançoire1; bouger balançoire2;`
  `bouger balançoire1; bouger balançoire2;...`

Deux peintres (vidéo)

Tiré de “Je ne sais rien mais je dirai tout”, Pierre Richard, 1973.

Explication

Victor Lanoux: **“tu dois compter jusqu’à trois, sinon tu perds un temps !”**

Un référentiel de temps logique absolu, partagé par tous

Il ne faut (surtout pas) chercher à aller le plus vite possible mais plutôt se mettre d’accord sur une échelle de temps commune

Respecter seulement le rythme du processus que l’on contrôle

```
loop                              loop
  bouger balançoire1;      ||       bouger balançoire2;
  pause                             pause
end                               end
```

On a appelé ça “Le modèle de temps synchrone” (Berry et al., 80’s)

Deux balancoires synchronisées

(Démo avec ReactiveML, Louis Mandel)

Est-ce si original que ça?

Le chef d'orchestre: tous les musiciens partagent un temps commun, celui du chef d'orchestre

Des danseurs: ils se synchronisent sur la musique. C'est la seule possibilité pour que plusieurs danseurs fassent la même chose.

Les circuits synchrones: ils se synchronisent sur les ticks d'une horloge.

On raisonne de manière idéale en négligeant la vitesse de la lumière (orchestre), du son (danseurs), de l'électricité (circuits).

On peut mesurer ces temps et vérifier qu'ils sont négligeables.

C'est tellement plus simple.

Ce temps global n’est pas nécessairement rapide

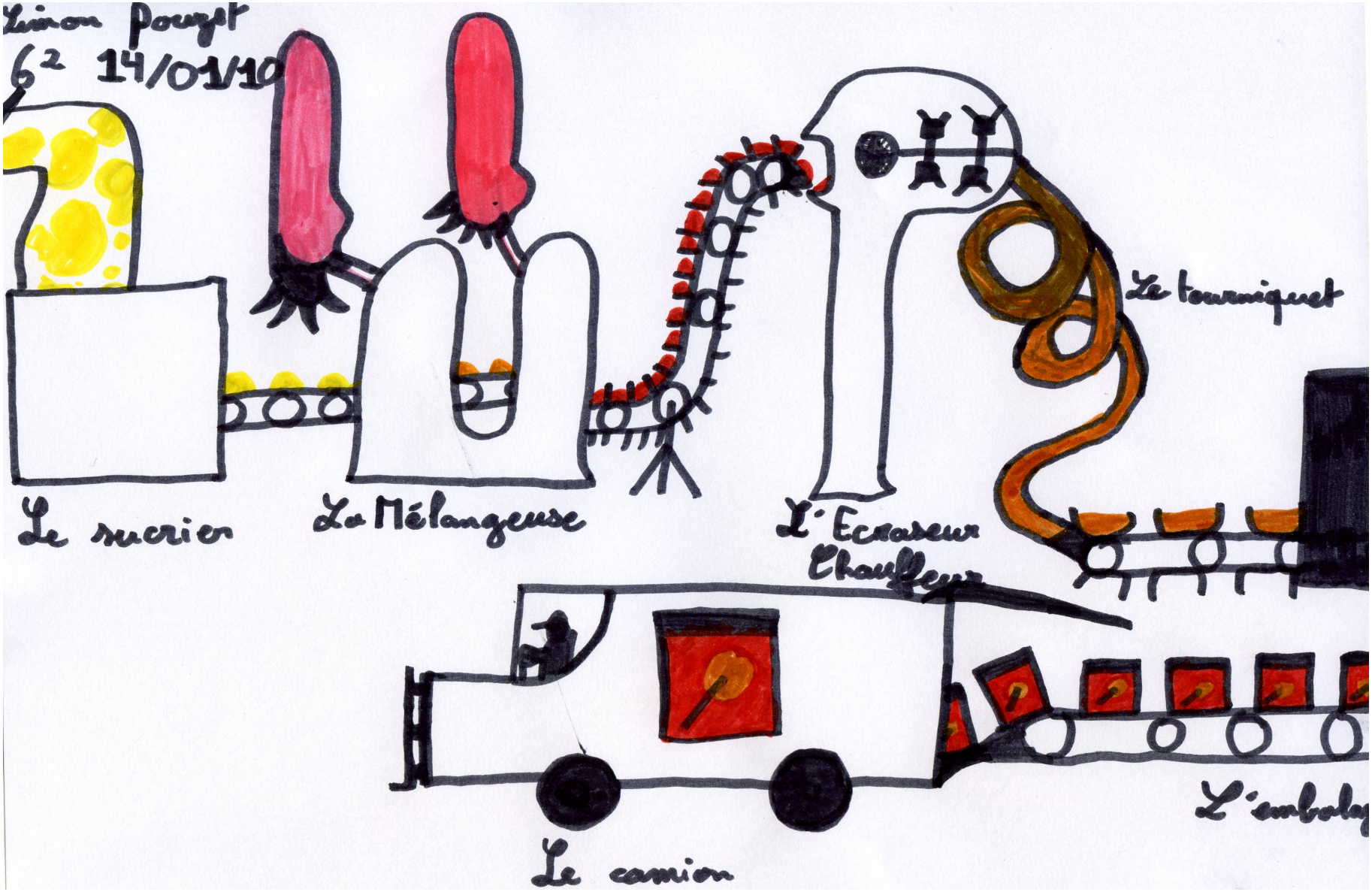
One time password

Par rapport au temps de saisie du mot de passe, la dérive sur les horloges est négligeable.

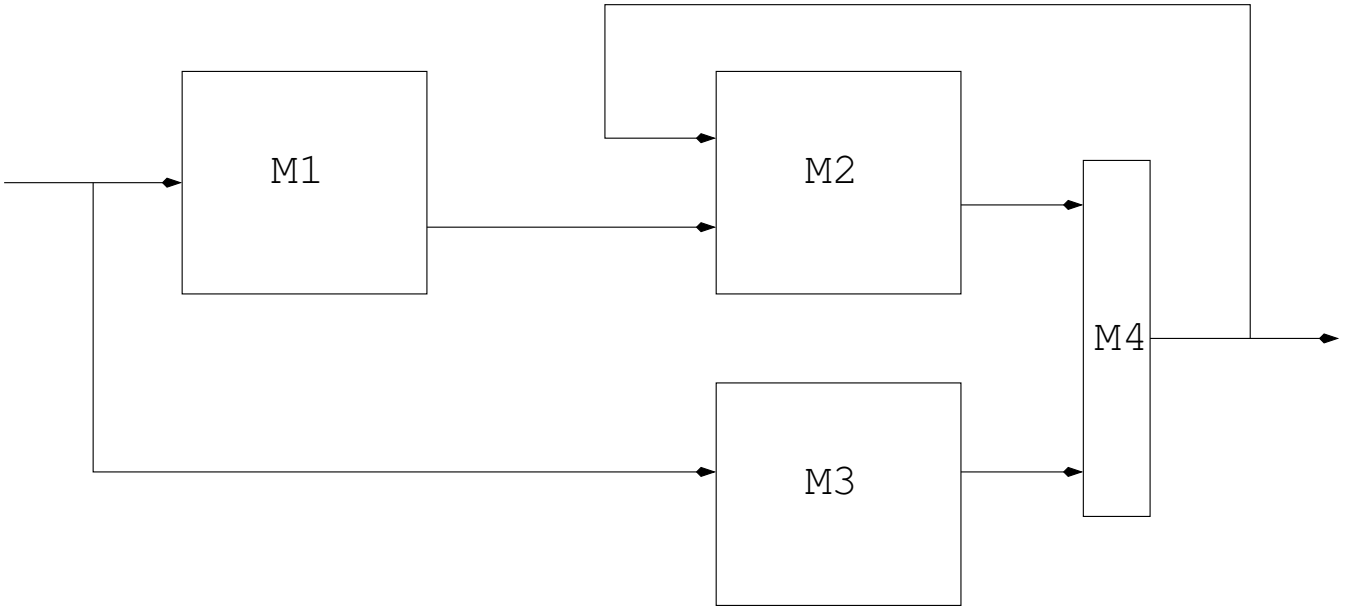
On peut donc faire “comme si” le calcul était instantané et se concentrer sur ce qui est important.

Ici il n’y a aucune synchronisation. On fait confiance au temps.

Des machines synchrones qui communiquent



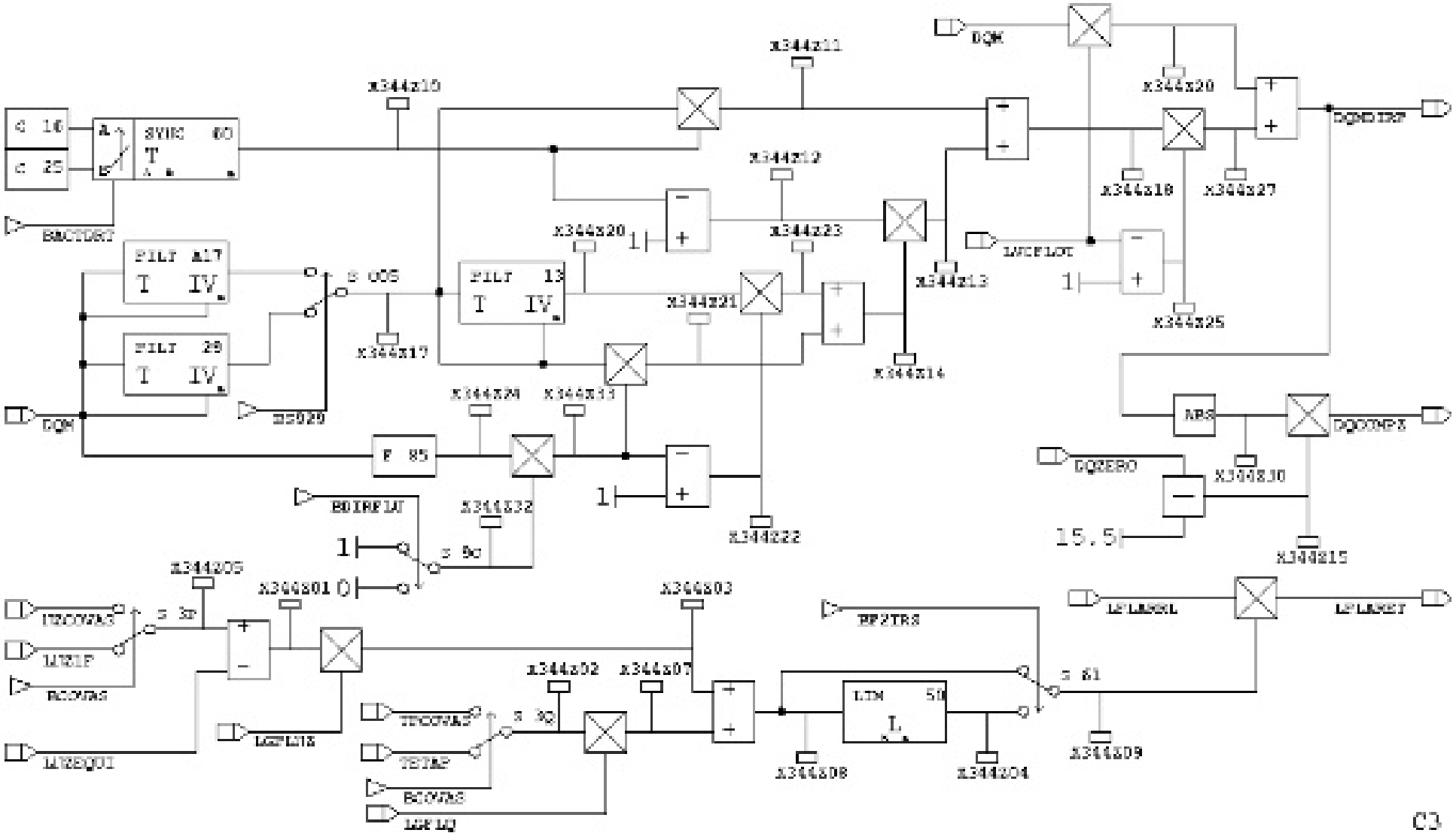
Réseaux de Kahn Synchrones



- des processus en parallèle qui communiquent par des flux de données
- communication en temps nul: une donnée produite est immédiatement disponible
- une notion de temps commun bien que les rythmes puissent être différents
- ces dessins ne sont pas loin des programmes informatiques

SAO (Spécification Assistée par Ordinateur) — Airbus 80's

Décrire le système sous forme de schémas/blocs (équations mathématiques)



Programmer en écrivant des équations de suites

Le langage Lustre (Caspi & Halbwachs, 1984).

X	1	2	1	4	5	6	...
Y	2	4	2	1	1	2	...
1	1	1	1	1	1	1	...
$X+Y$	3	6	3	5	6	8	...
$X+1$	2	3	2	5	6	7	...

L'équation $Z = X + Y$ signifie qu'à tout instant n, $Z_n = X_n + Y_n$.

Le temps est logique: les deux entrées X et Y arrivent "en même temps"; la sortie Z est produite au même instant.

En pratique, il suffit de vérifier que la sortie à l'instant courant est produite avant l'entrée arrivant à l'instant suivant.

Mémoriser la valeur d'une entrée

On ajoute des opérateurs pour mémoriser une entrée à l'instant précédent.

`X`	1	2	1	4	5	6	...
`pre X`	nil	1	2	1	4	5	...
`Y`	2	4	2	1	1	2	...
`Y -> pre X`	2	1	2	1	4	5	...
`S`	1	3	4	8	13	19	...

La suite (S_n) telle que $S_0 = X_0$ et $S_n = S_{n-1} + X_n$ pour $n > 0$ s'écrit:

```
S = X -> pre S + X
```

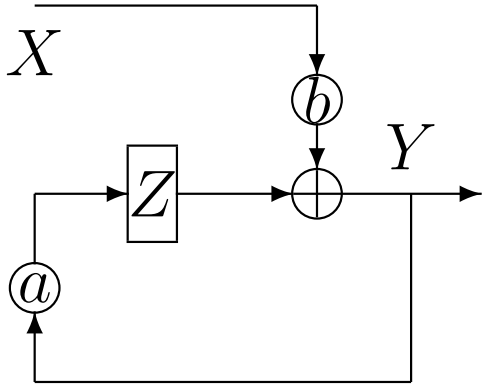
Comme en mathématique, on peut introduire des équations intermédiaires:

```
S = X -> I; I = pre S + X
```

Un modèle classique de l'automatique et de l'électronique

Exemple: un filtre linéaire

$$Y_0 = bX_0 \ , \ \forall n \ Y_{n+1} = aY_n + bX_{n+1}$$



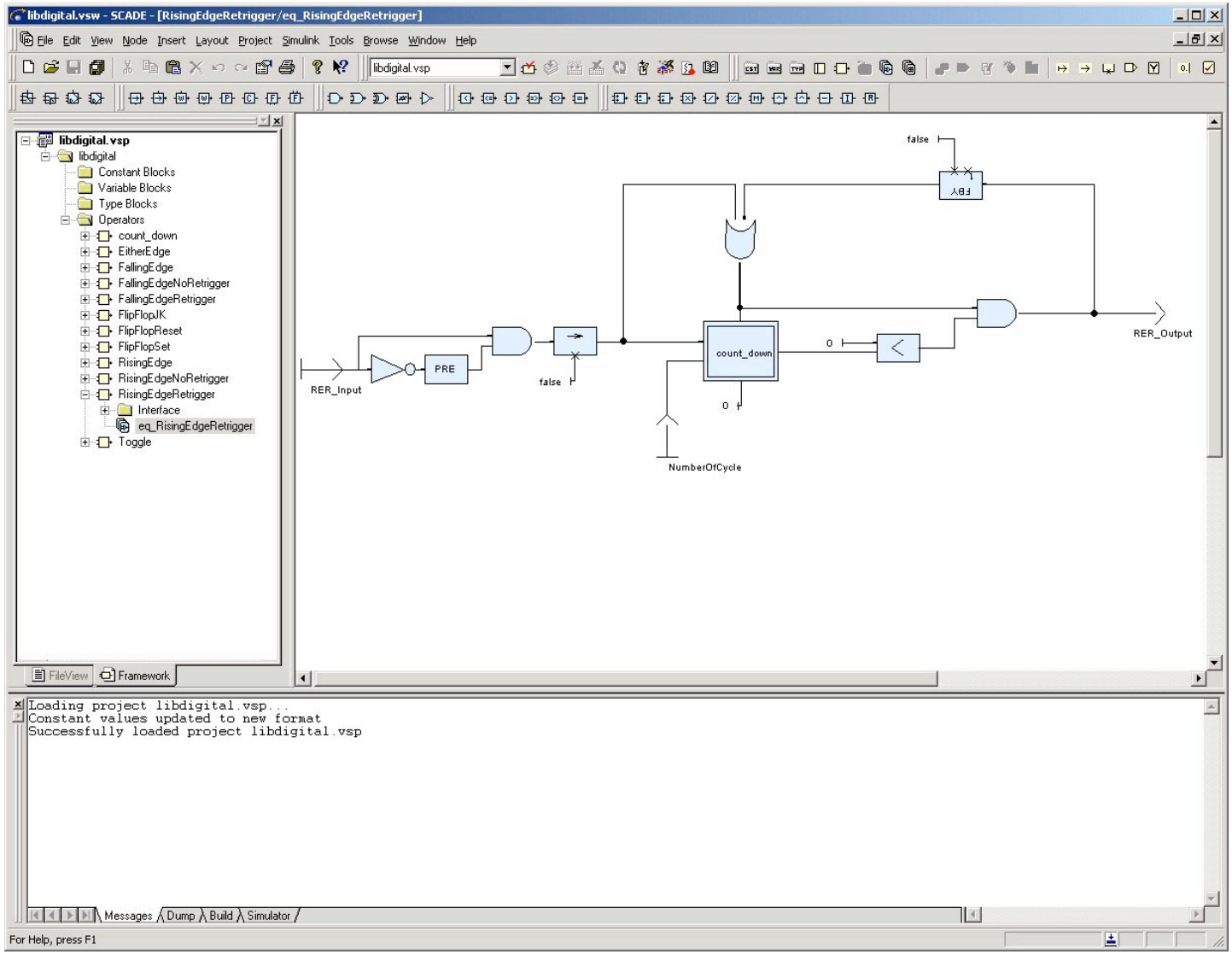
(démo — filtre linéaire en Lustre)

L'idée de Lustre:

- écrire directement des équations mathématiques
- les analyser, les transformer
- les traduire automatiquement en programmes informatiques exécutables

SCADE (Safety Critical Application Development Environment – Esterel-Tech.)

Du dessin par ordinateur à des programmes exécutables



Mélanger des rythmes différents

(Vidéo – briques)

Une seule brique à la fois; elle ne doivent pas arriver trop vite...

- $\mathtt{G_i} = 1$ lorsque la brique est à ma gauche
- $\mathtt{D_i} = 1$ lorsque la brique est à ma droite

$\mathtt{G_1}$	1	0	0	0	0	0	...
$\mathtt{D_1}$	0	1	0	0	0	0	...
$\mathtt{G_2}$	0	1	0	0	0	0	...
$\mathtt{D_2}$	0	0	1	0	0	0	...
$\mathtt{G_3}$	0	0	1	0	0	0	...
$\mathtt{D_3}$	0	0	0	1	0	0	...

Mélanger des rythmes différents

(Vidéo – briques)

Une seule brique à la fois; elle ne doivent pas arriver trop vite...

- $\mathtt{G_i} = 1$ lorsque la brique est à ma gauche
- $\mathtt{D_i} = 1$ lorsque la brique est à ma droite

$\mathtt{G_1}$	1	0	1	0	0	0	...
$\mathtt{D_1}$	0	1	0	1	0	0	...
$\mathtt{G_2}$	0	1	0	1	0	0	...
$\mathtt{D_2}$	0	0	1	0	1	0	...
$\mathtt{G_3}$	0	0	1	0	1	0	...
$\mathtt{D_3}$	0	0	0	1	0	1	...

Les synchroniser

(Vidéo – briques)

Tiré de “Je ne sais rien mais je dirai tout”, Pierre Richard, 1973.

Que s’est-il passé?

Deux rythmes différents non synchronisables !

1er cas:

- Pierre Richard ne sait faire qu’une chose à la fois ($\mathtt{C}$ (tirer la corde) ou G_6 (brique à gauche))
- La corde doit toujours être tenue, i.e., $\mathtt{C} = 1$

2ième cas:

- Il fait deux choses en même temps (en somme, il va deux fois plus vite)
- Cette fois, ça marche... presque

Est-ce si important d'être parfaitement synchrone?

Vérifier les rythmes

- Trouver des méthodes pour vérifier que les rythmes sont corrects
- Une fois les rythmes connus, établir une fois pour toute qui doit travailler et quand

Même si les rythmes sont bons et tombent en face; il faut encore que la brique arrive au bon moment.

Solution: une table ou poser la brique (buffer). Cela va introduire un délai (latence)

C'est exactement ce qui se passe dans une télévision numérique (TNT) *vs* une télévision analogique: la première image arrive avec une seconde de délai.

On utilise des buffers pour synchroniser l'image et le son.

Conclusion

Les langages synchrones ont été inventés pour concevoir/programmer les systèmes réactifs temps réel. Ils sont utilisés en grand (avions, métros, trains, etc.).

Modèle du temps synchrone

- une notion de **temps logique commun à tous** les processus
- **raisonner idéalement** par rapport à celui-ci
- vérifier **a posteriori** que l'implémentation est suffisamment rapide

Programmer en écrivant des équations mathématiques

- un système = un ensemble d'équations sur des suites
- correspond au modèle de l'automatique/électronique/traitement du signal
- ces équations mathématiques sont traduites automatiquement en code exécutable, réduisant ainsi le risque de bogue