Parallélisme Synchrone

Gérard Berry

Collège de France
Chaire Informatique et sciences numériques

Cours 6 du 13 janvier 2010

Parallélismes synchrone et vibratoire

Synchrone

Musiciens et spectateurs négligent la vitesse du son

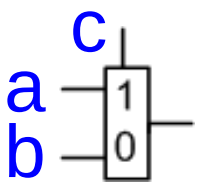
Vibratoire

Mais les acousticiens règlent sa propagation

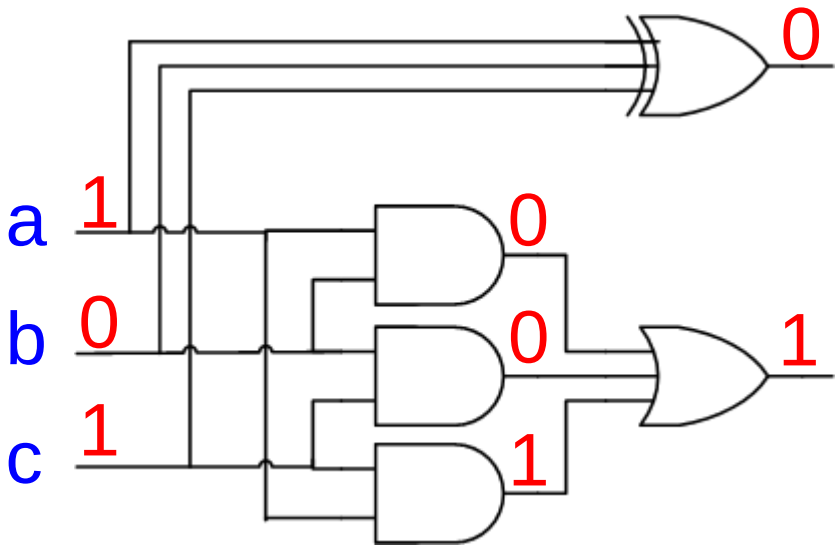
Agenda

1. Circuits digitaux
2. Des automates aux circuits
3. Esterel et SyncCharts
4. Sémantique
5. Traduction en circuits
6. Optimisation
7. Circuits cycliques

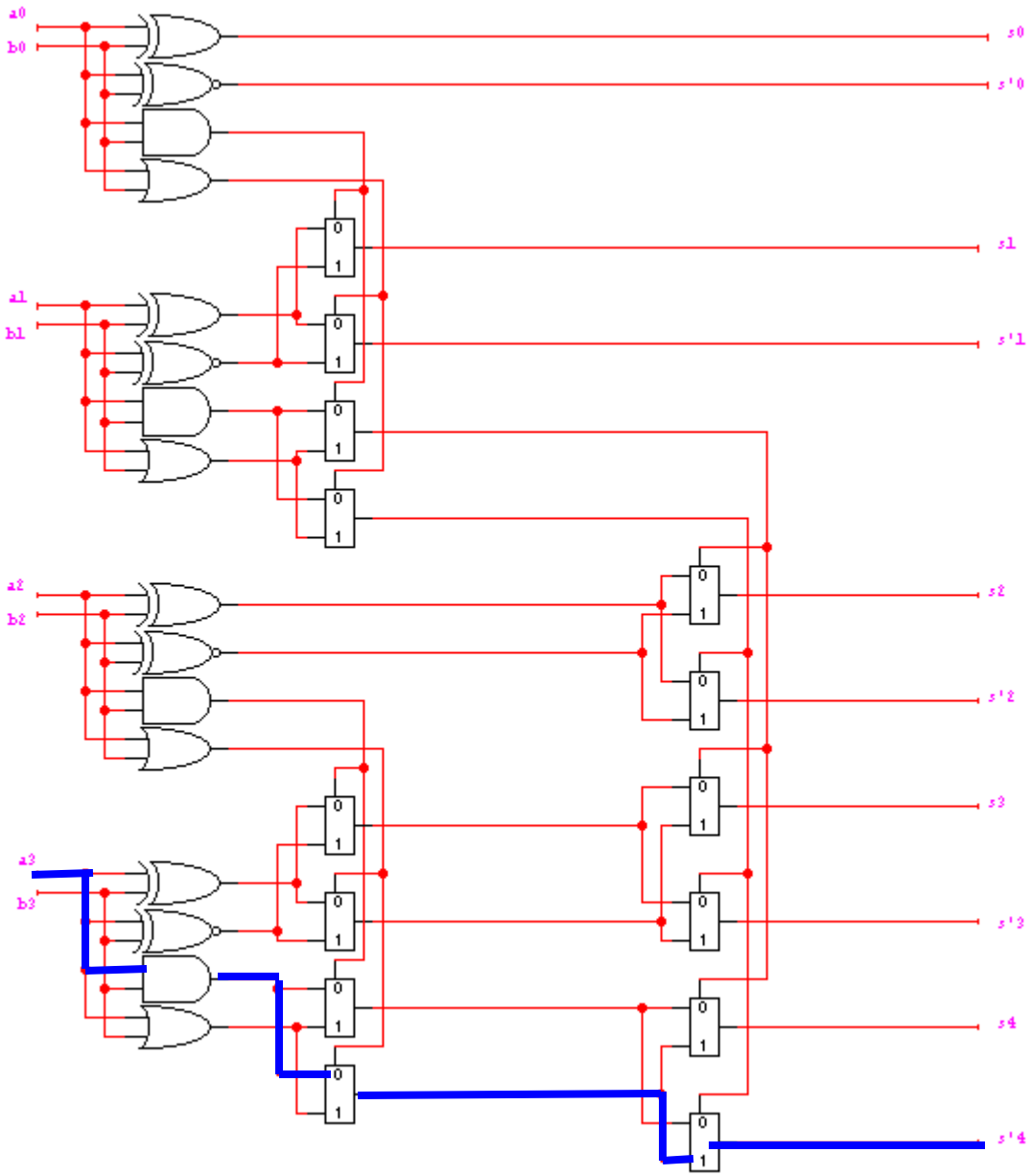
Circuits digitaux combinatoires



mux(c,a,b) =(c et a) ou ((non c) et b)

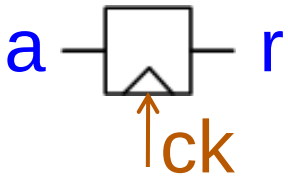


Additionneur de von Neumann

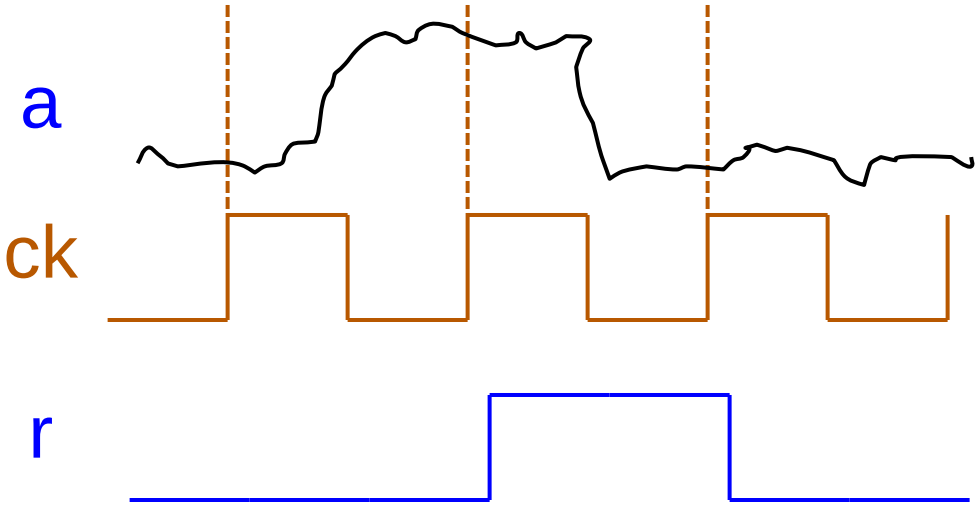


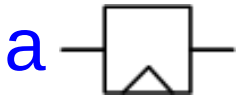
Pour n bits
temps log(n)

Le registre

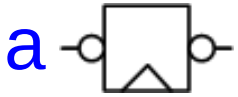


reg(a)



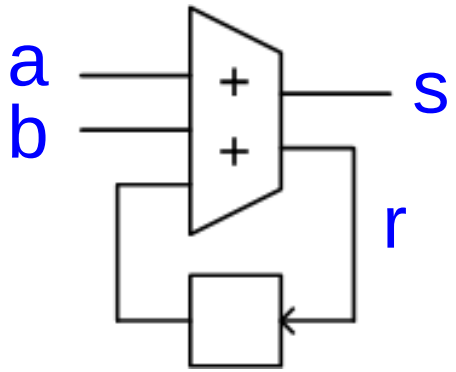


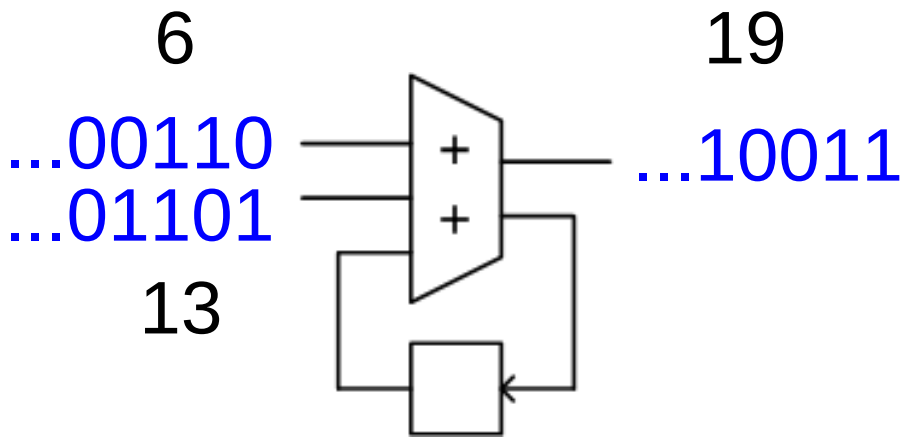
$a = a_0, a_1, a_2, \ldots$ $r = 0, a_0, a_1, a_2, \ldots$



$a = a_0, a_1, a_2, \ldots$ $r = 1, a_0, a_1, a_2, \ldots$

Les circuits séquentiels



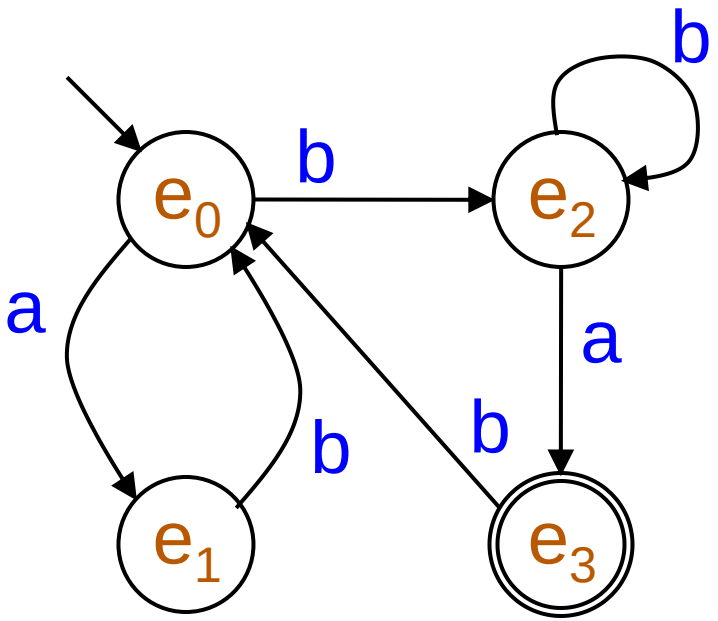


additionneur sériel

Agenda

1. Circuits digitaux
2. Des automates aux circuits
3. Esterel et SyncCharts
4. Sémantique
5. Traduction en circuits
6. Circuits cycliques

(ab+b)*ba

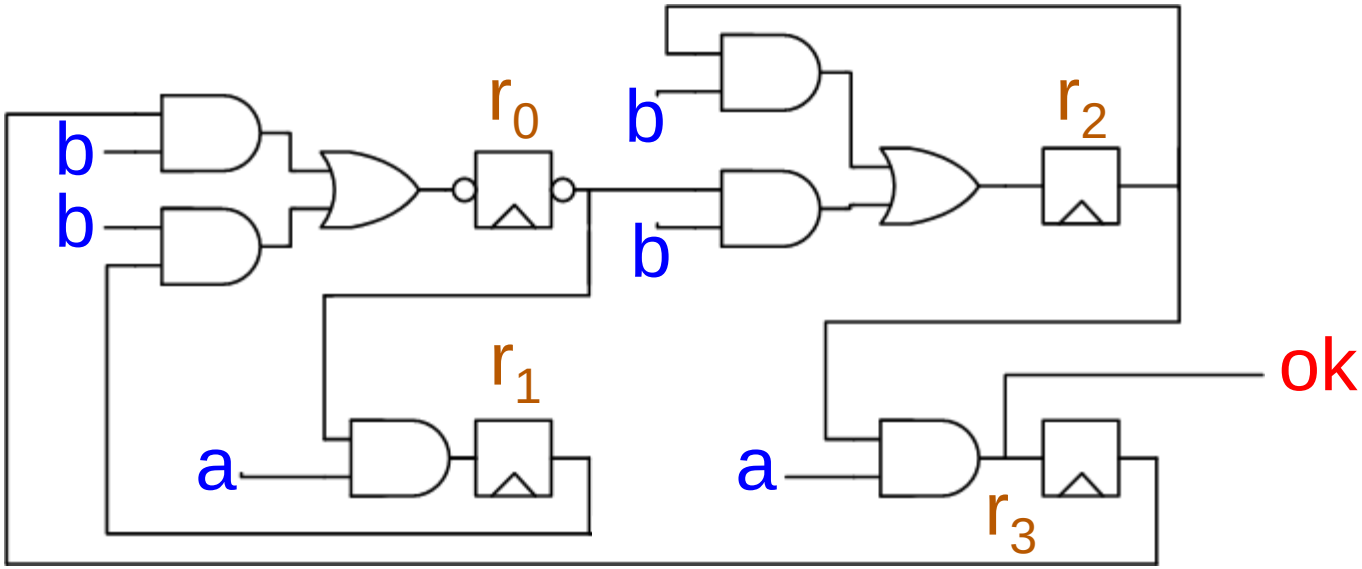


déterministe

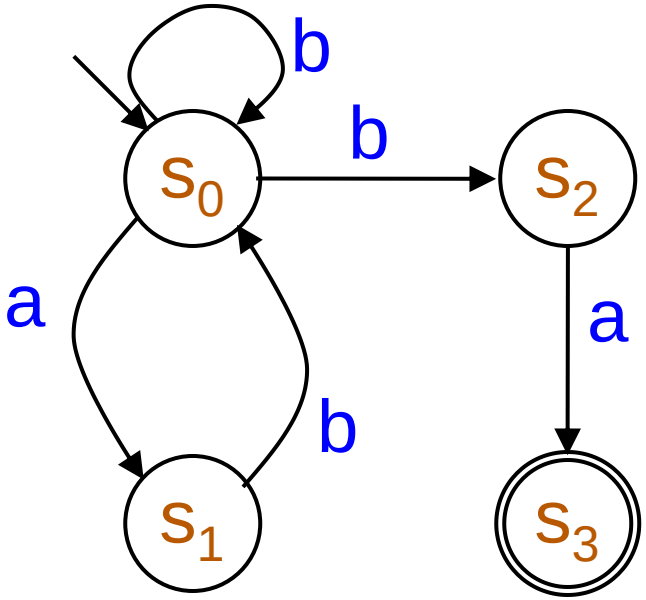
⇒1-hot

(1 seul r_i à 1)

explosion en taille !



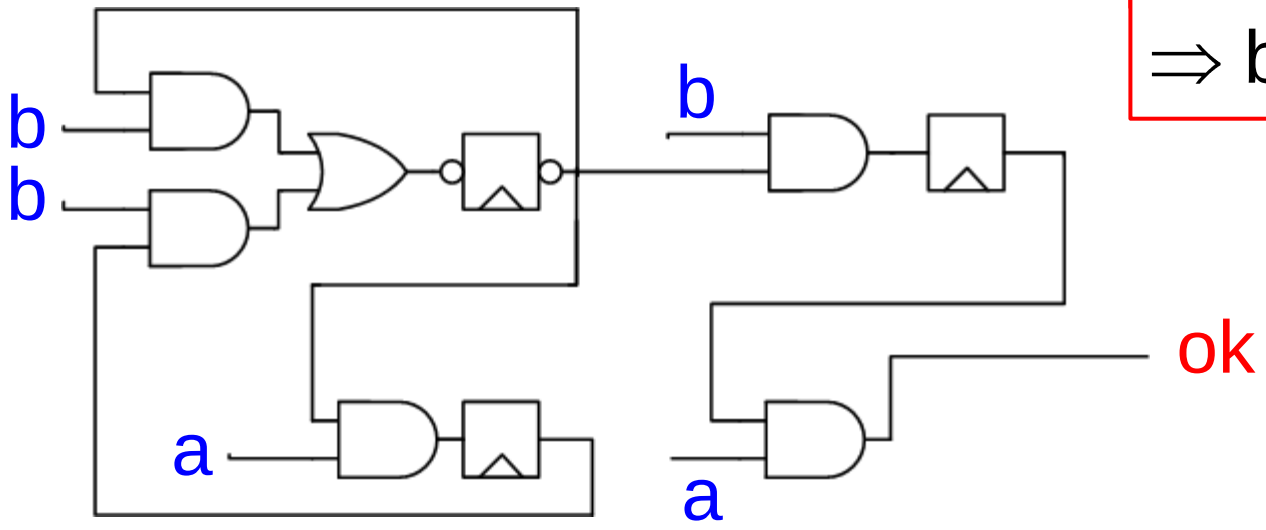
(ab+b)*ba



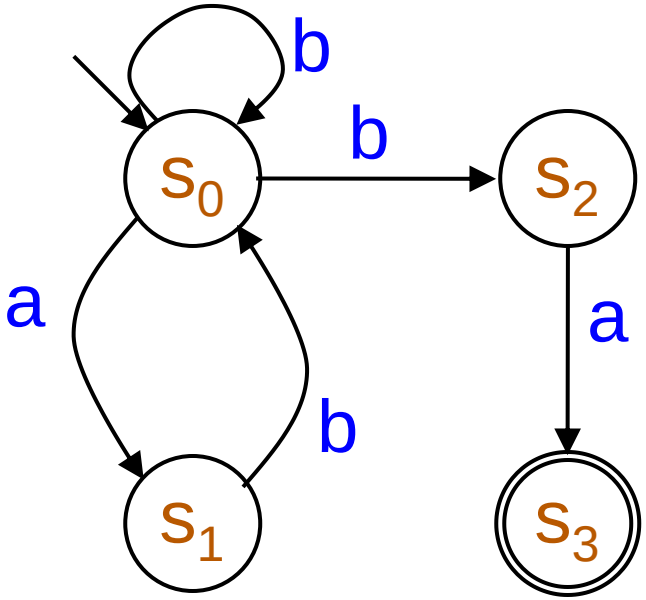
non-déterministe

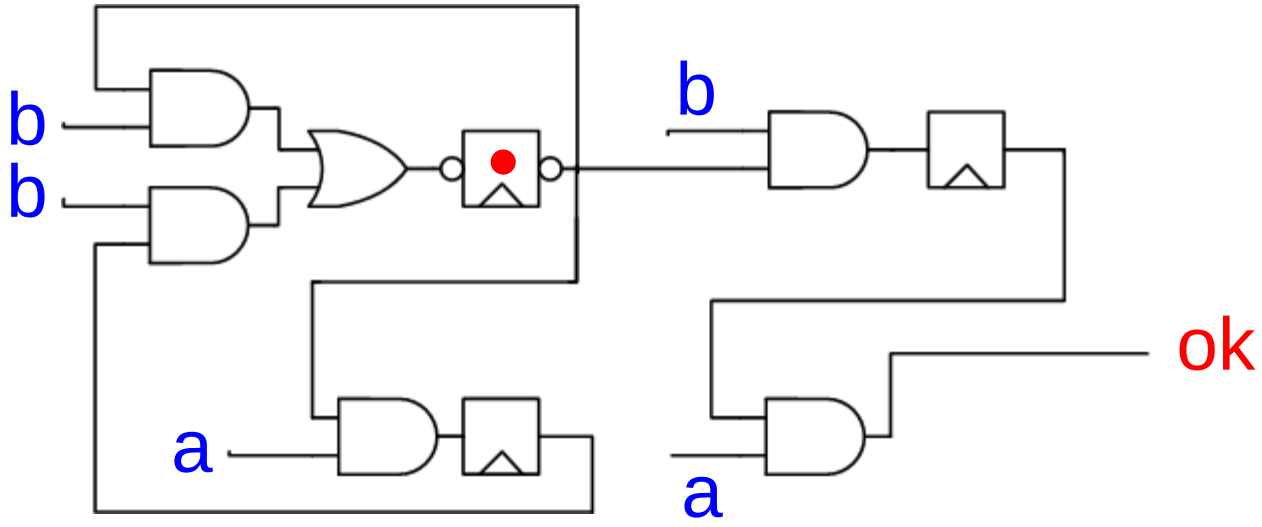
⇒ pas d'explosion

⇒ bien meilleur !

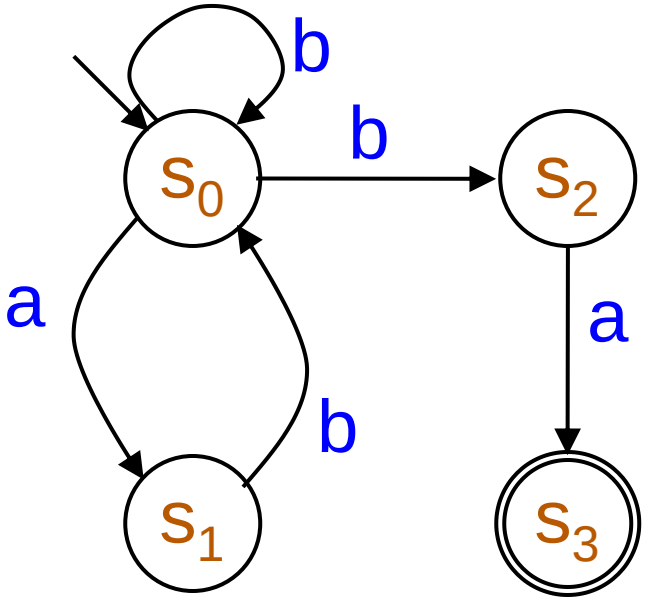


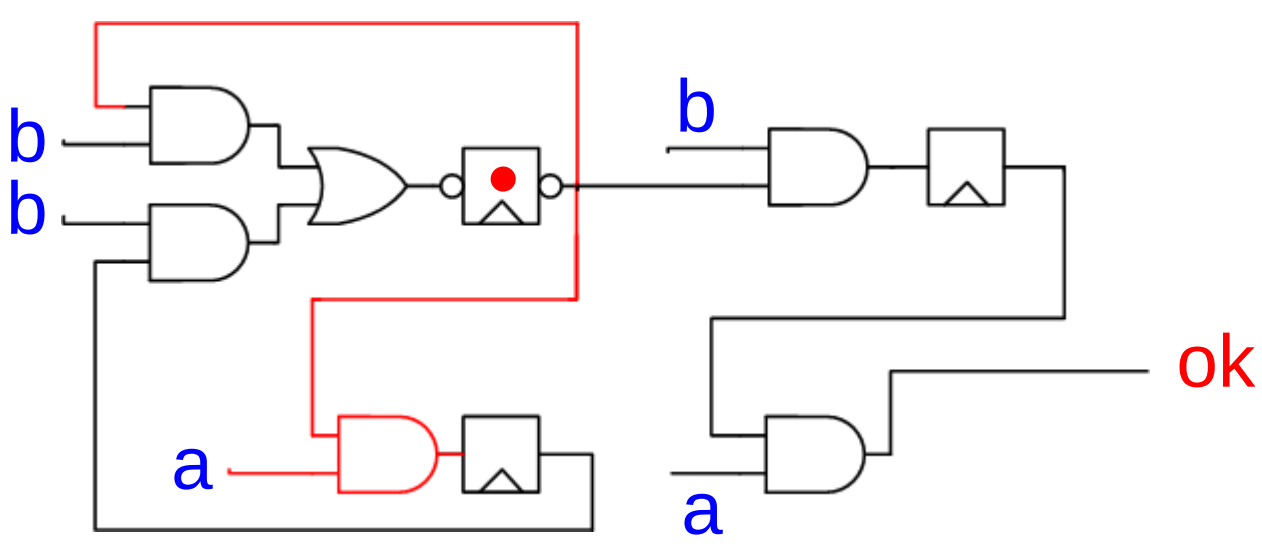
(ab+b)*ba





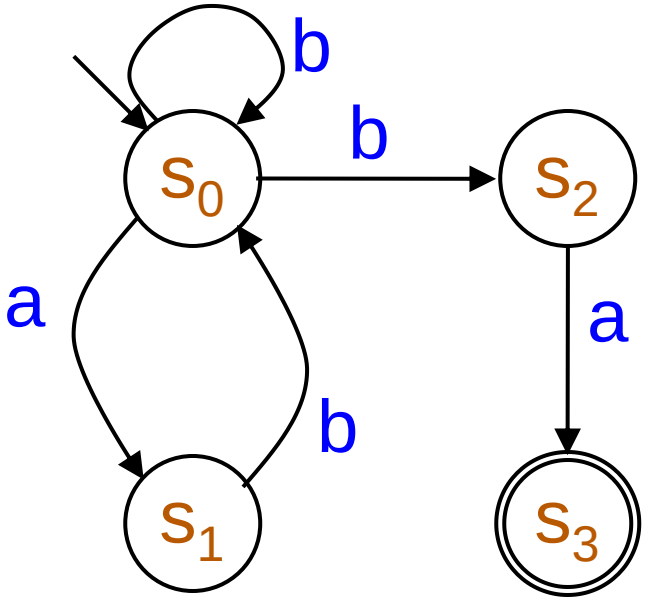
(ab+b)*ba

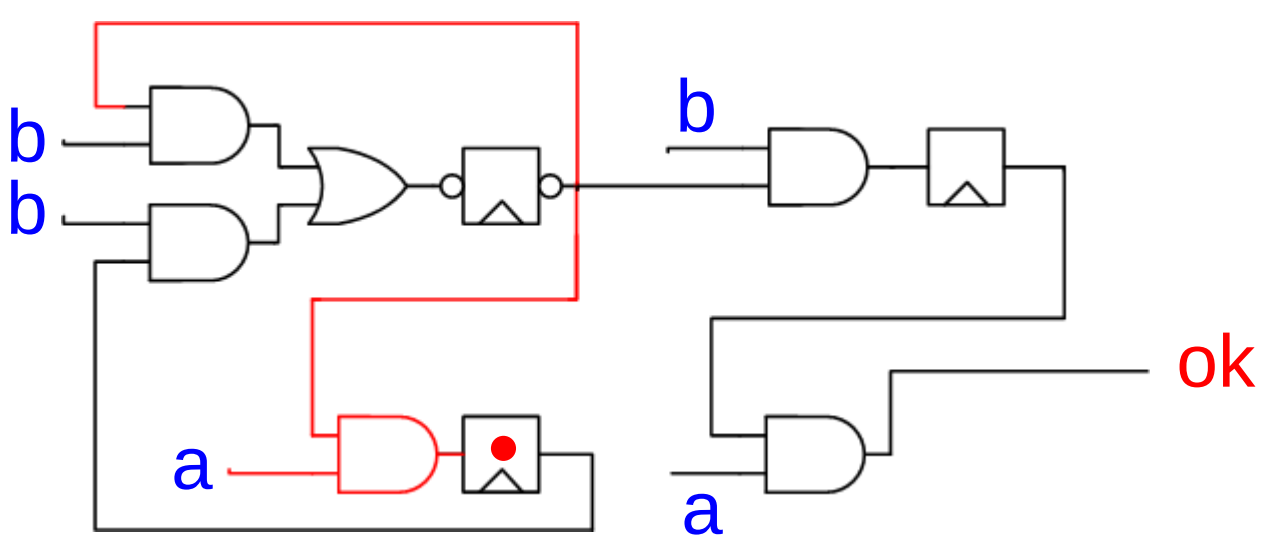




a

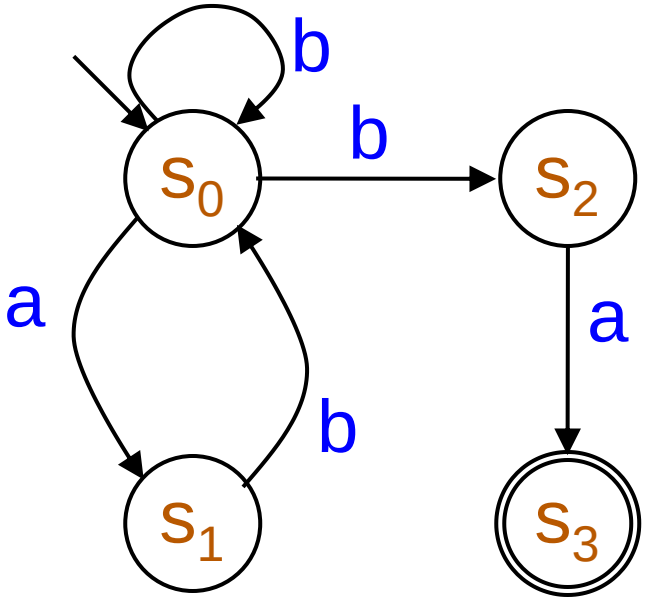
(ab+b)*ba

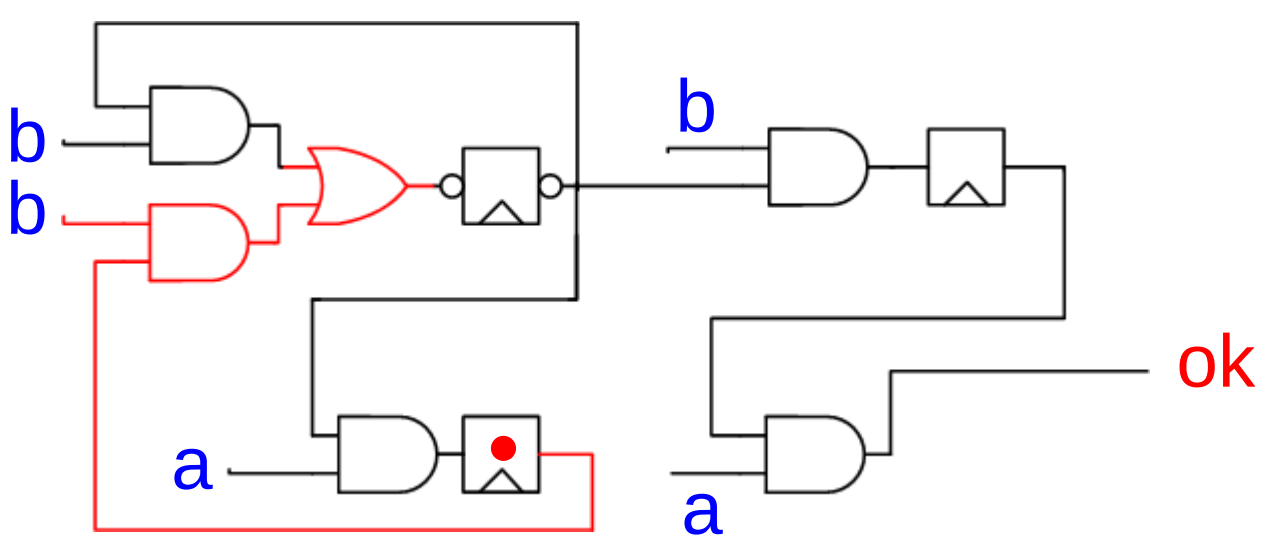




a

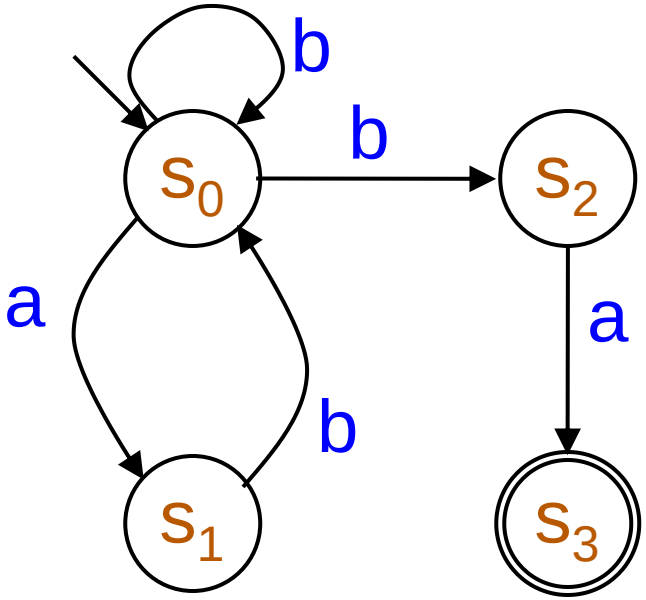
(ab+b)*ba

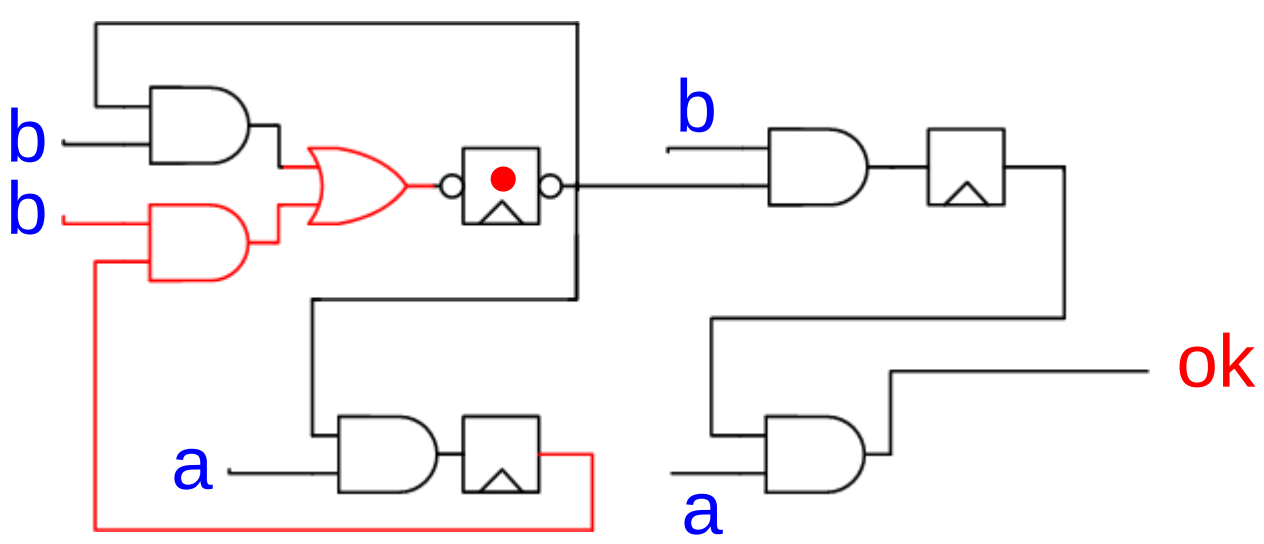




ab

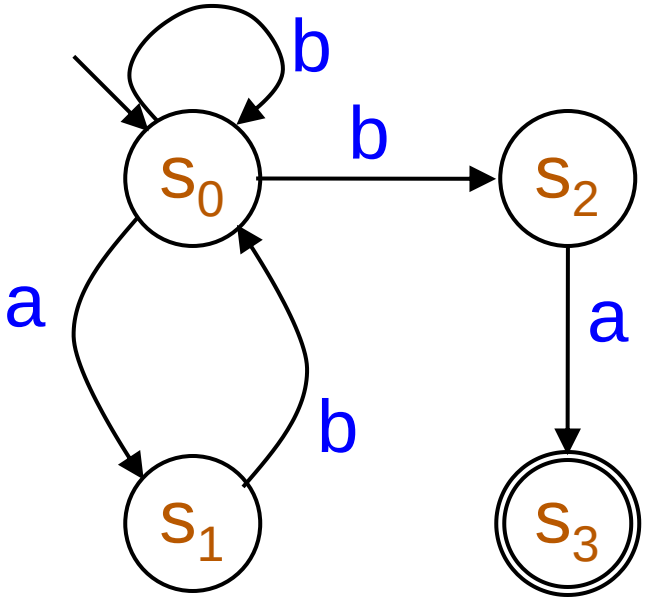
(ab+b)*ba

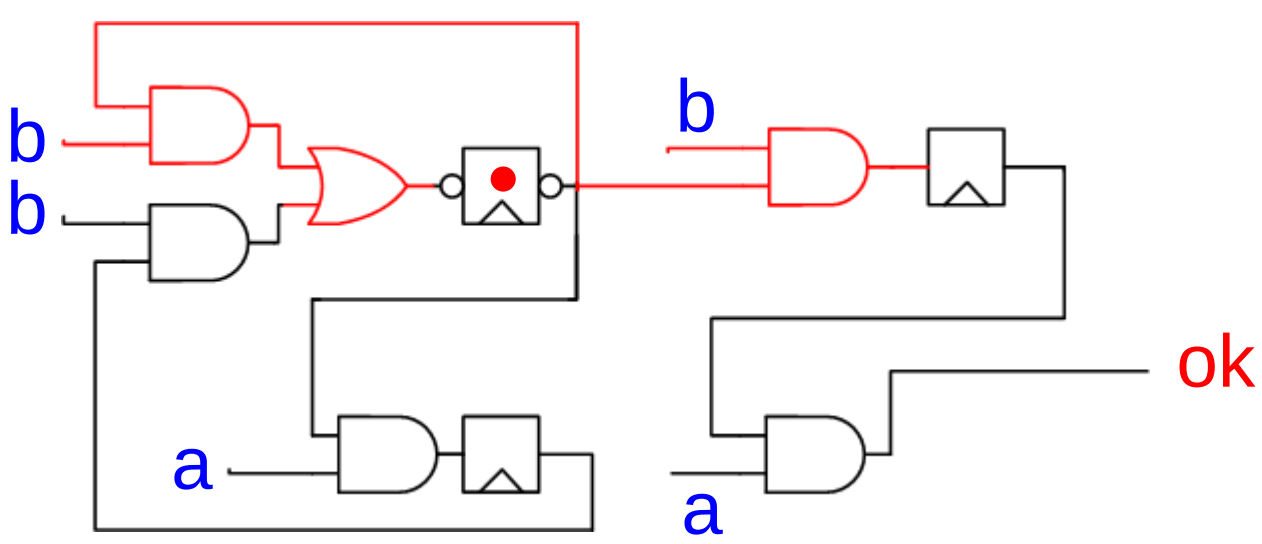




ab

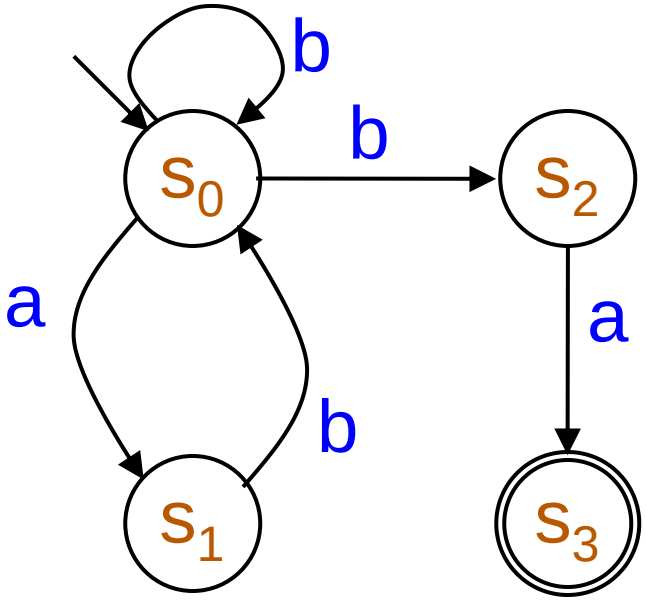
(ab+b)*ba

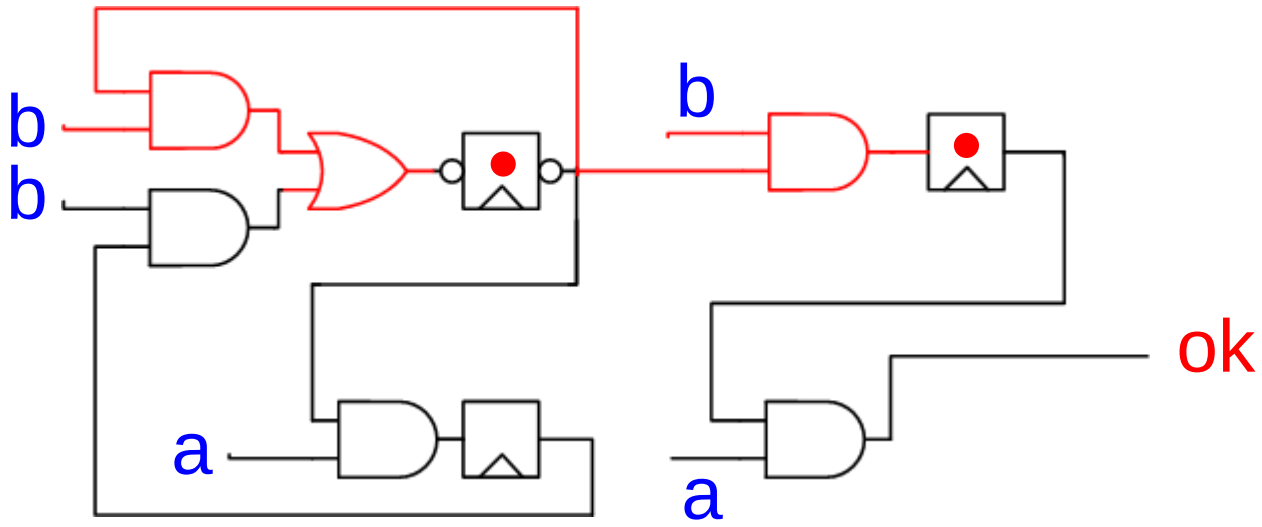




abb

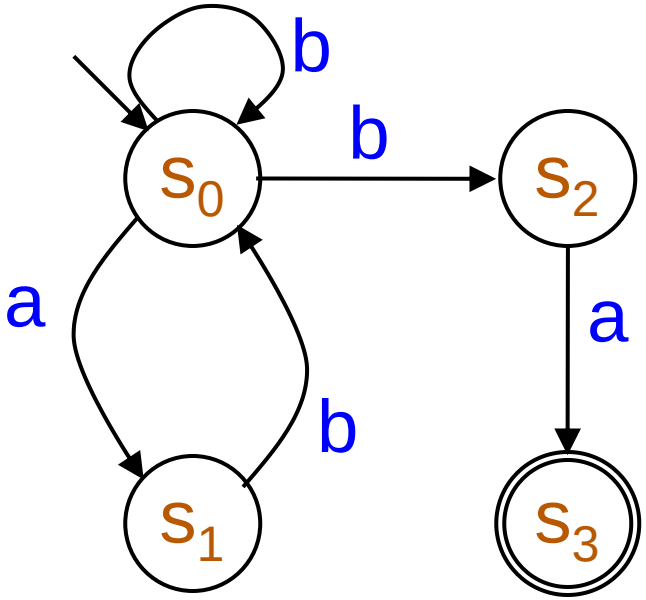
(ab+b)*ba

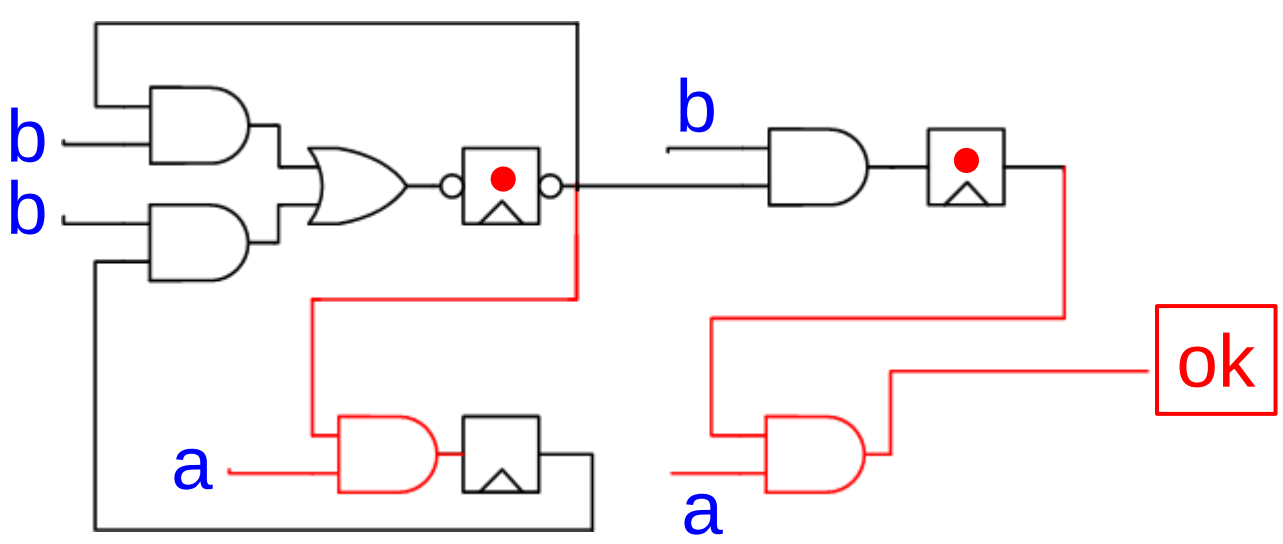




abb

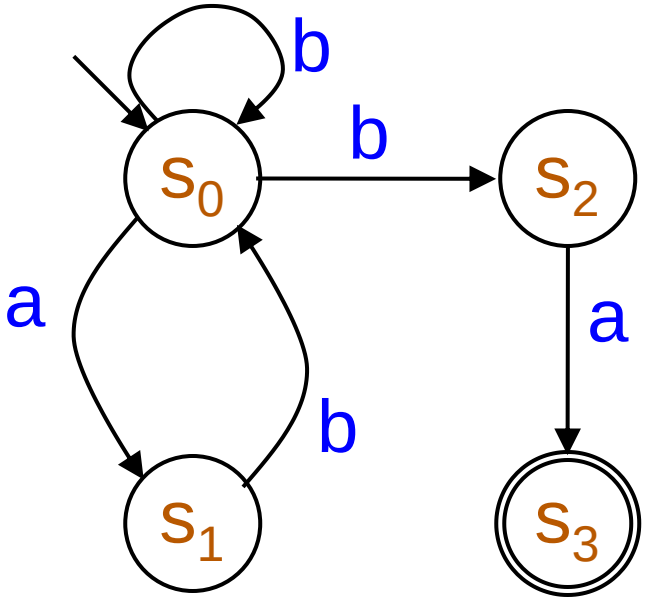
(ab+b)*ba

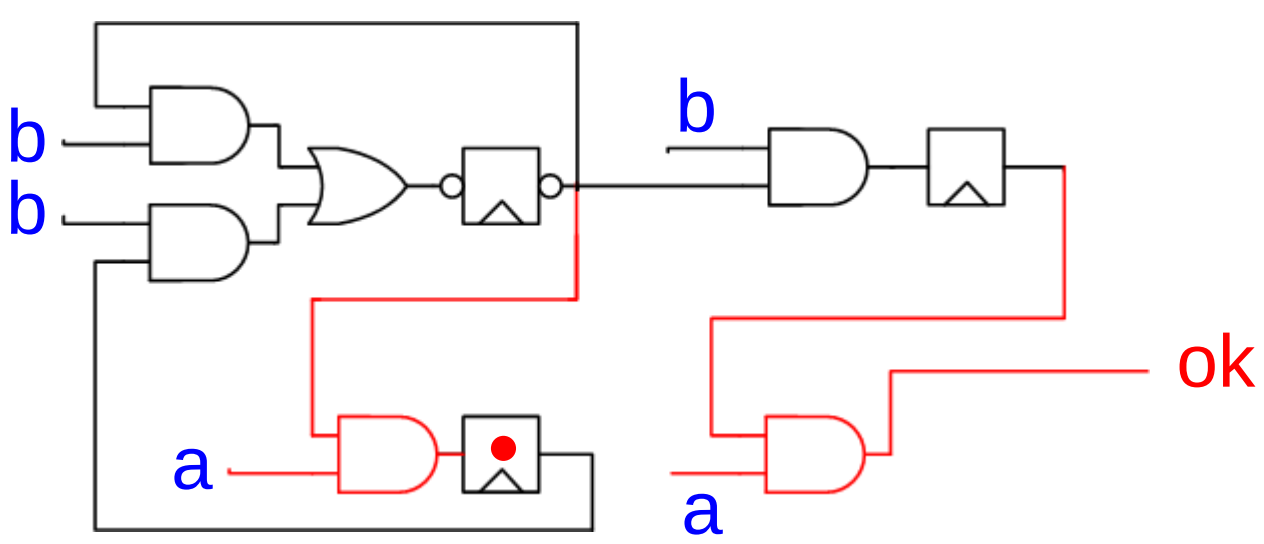




abba

(ab+b)*ba





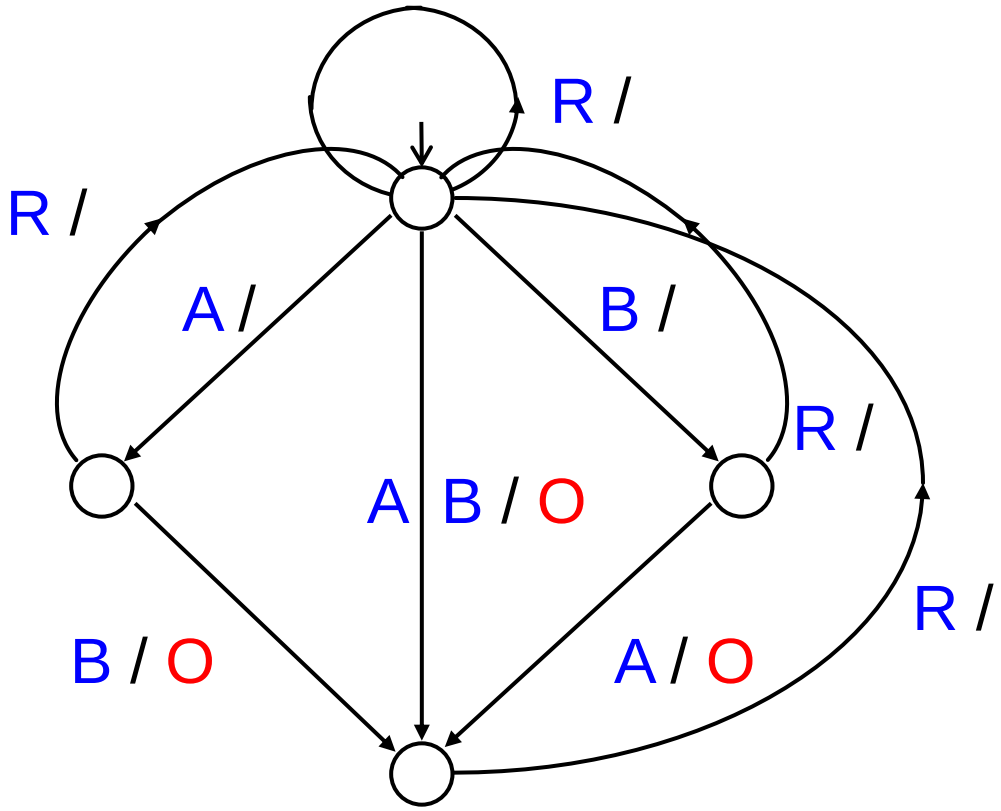
abba

Agenda

1. Circuits digitaux
2. Des automates aux circuits
3. Esterel et SyncCharts
4. Sémantique
5. Traduction en circuits
6. Optimisation
7. Circuits cycliques

L'exemple ABRO

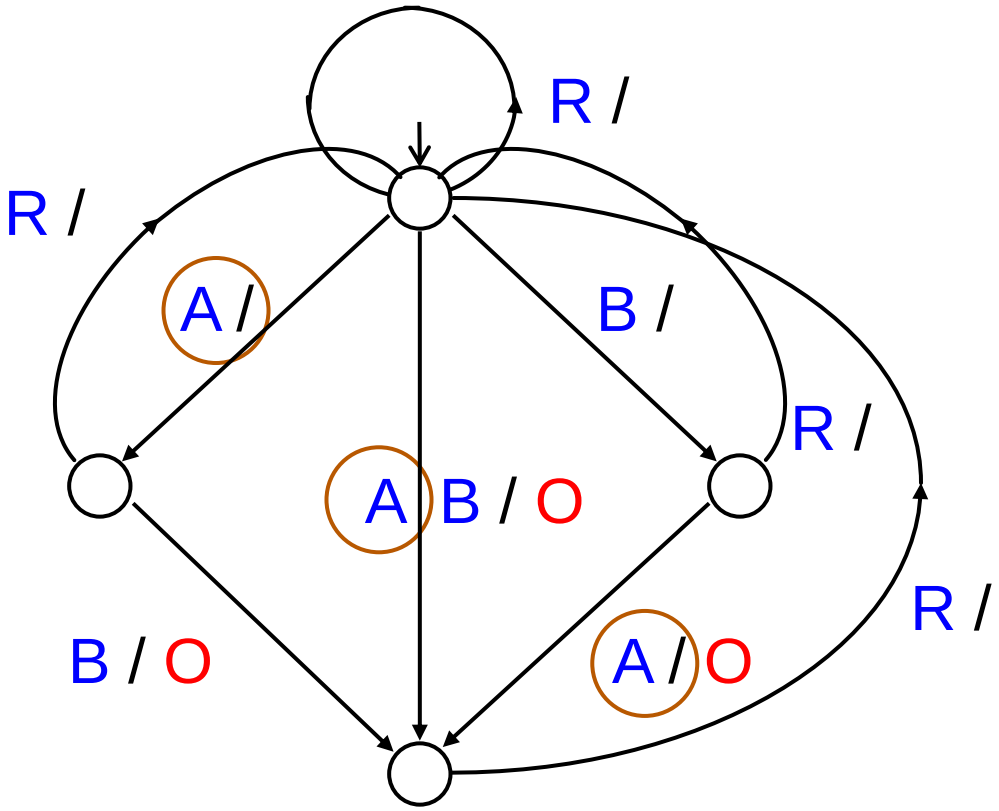
Emettre O dès que A and B sont arrivés
Réinitialiser le comportement à chaque R



Ecriture mémoire
R : demande
A : adresse
B : donnée
O : écriture

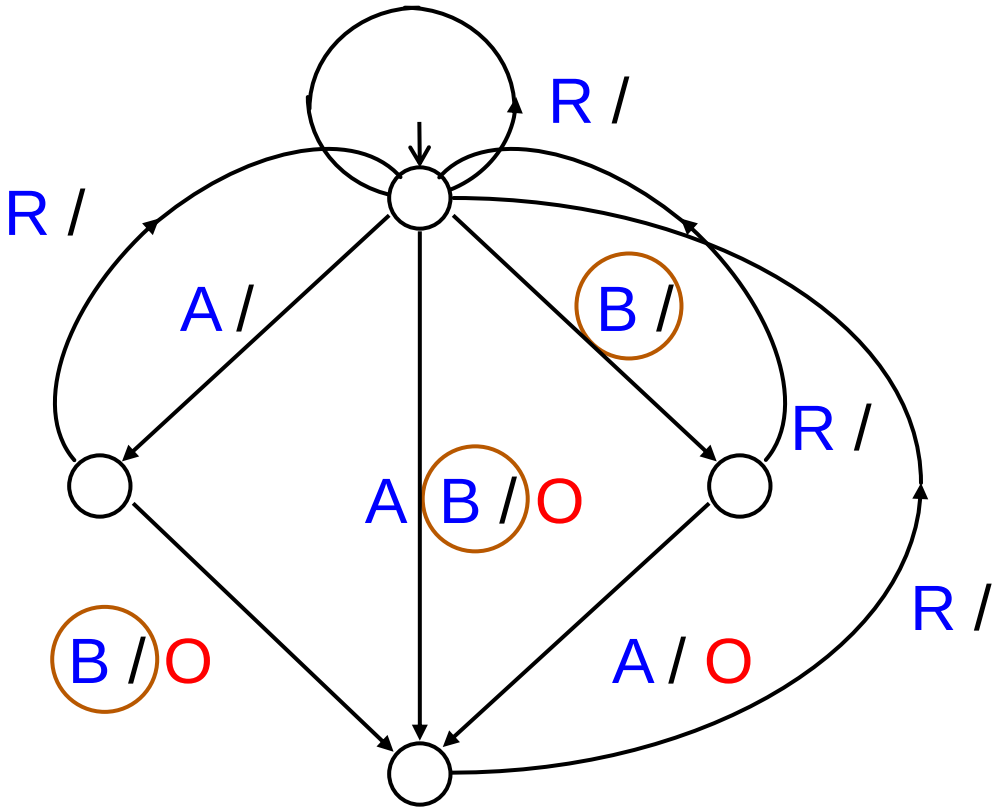
L'exemple ABRO

Emettre O dès que A and B sont arrivés
Réinitialiser le comportement à chaque R



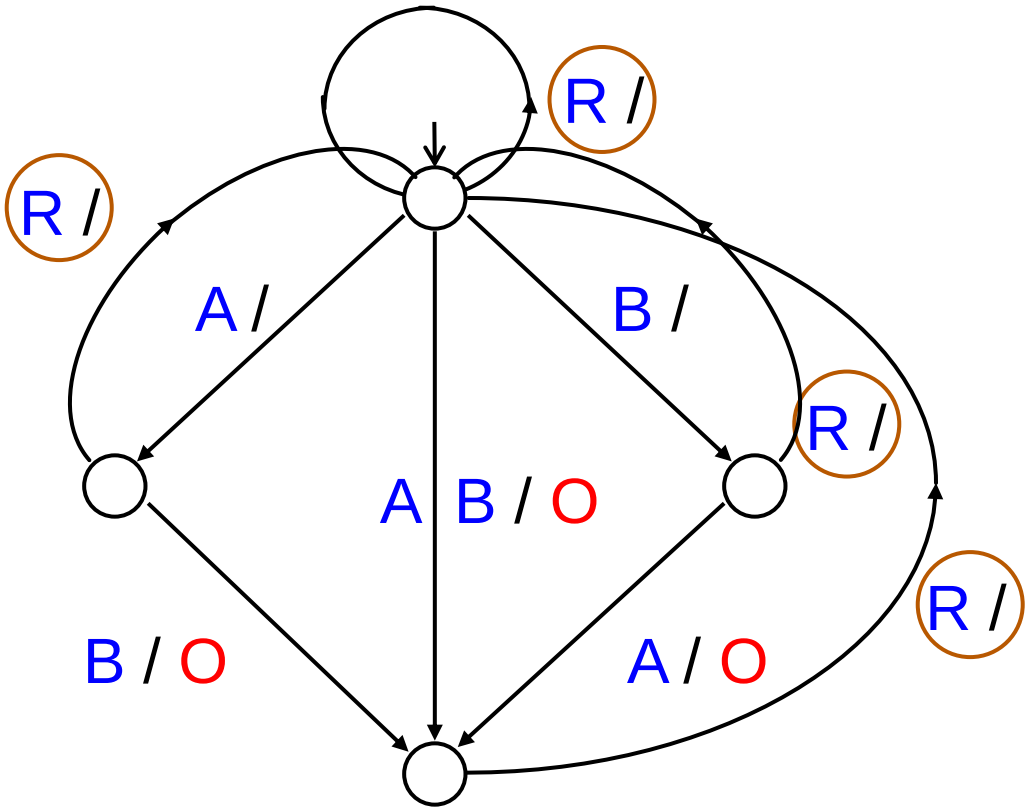
L'exemple ABRO

Emettre O dès que A and B sont arrivés
Réinitialiser le comportement à chaque R



L'exemple ABRO

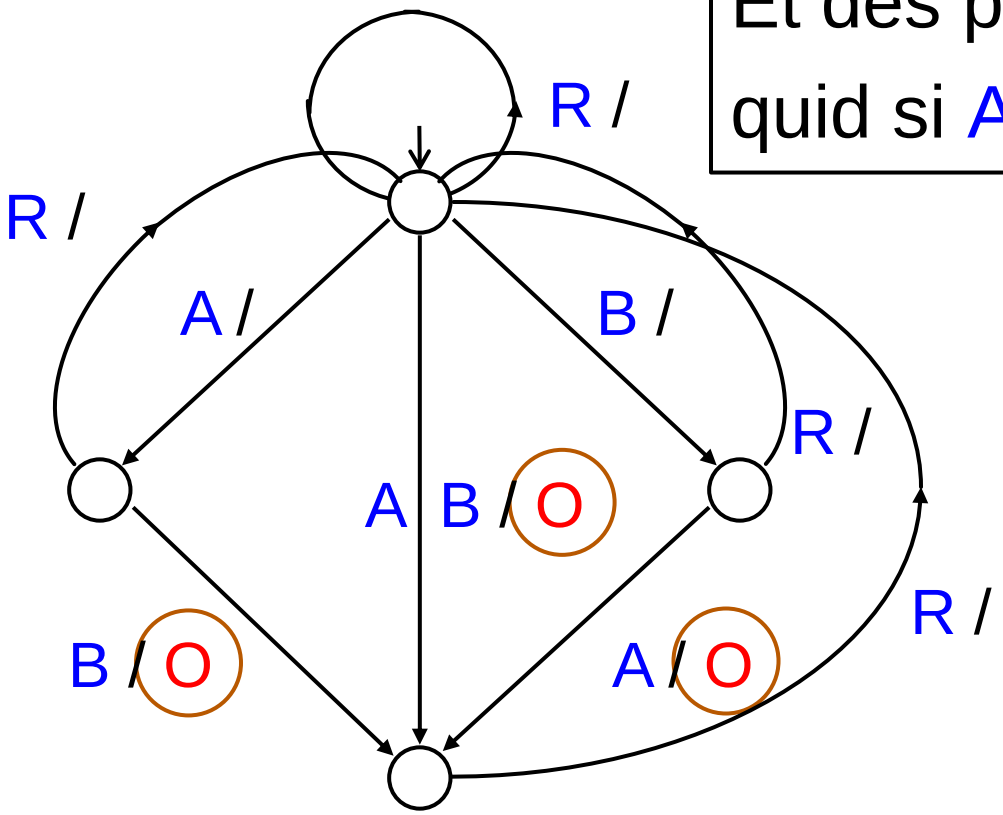
Emettre O dès que A and B sont arrivés
Réinitialiser le comportement à chaque R



L'exemple ABRO

Emettre O dès que A and B sont arrivés
Réinitialiser le comportement à chaque R

Et des problèmes de priorité :
quid si A, B, R ensemble?



~~Programmer par couper / coller~~

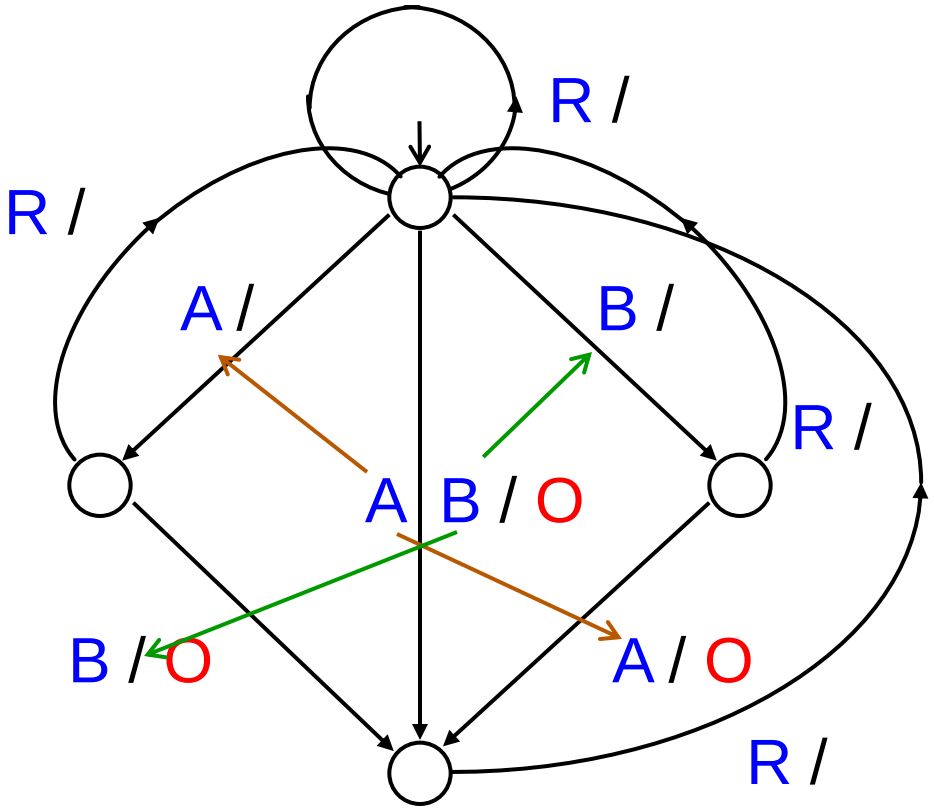
Ecrire chaque chose une fois et une seule !

~~Expressions rationnelles~~

Automates hiérarchiques
(Statecharts, SyncCharts, Stateflow)

Langages synchrones impératifs
Esterel, Quartz

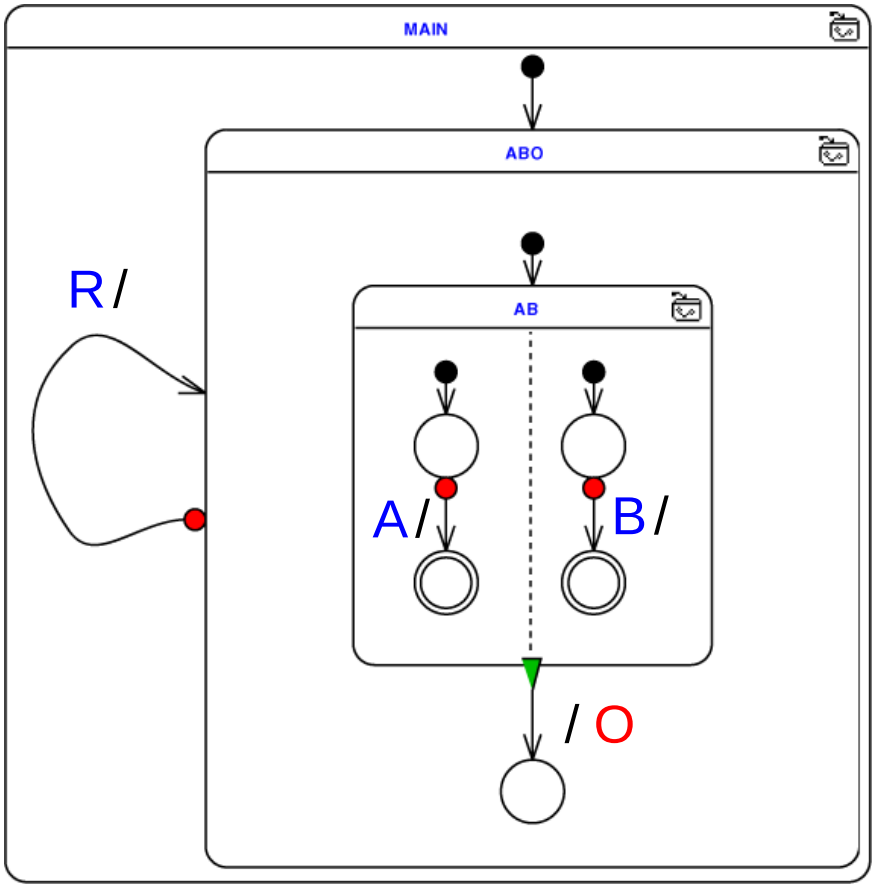
Esterel = spécification linéaire




```
loop
   abort
        { await A || await B };
        emit O;
        halt
   when R
end loop
```

copies = résidus !
Esterel = partage des résidus

SyncCharts (C. André)




```
loop
   abort
        { await A || await B };
        emit O;
        halt
   when R
end loop
```

automates parallèles
hiérarchiques synchrones

Le coureur Esterel

```
trap HeartAttack in
    every Morning do
        abort
            loop
                abort  run Slowly  when  100 Meter ;
                abort
                    every Step do
                        run Jump || run Breathe || run CheckHeart
                    end every
                when 15 Second ;                    exit HeartAttack
                run FullSpeed
            each  Lap
        when  4 Lap
    end every
handle HeartAttack fo
   run RushToHospital
end trap
```

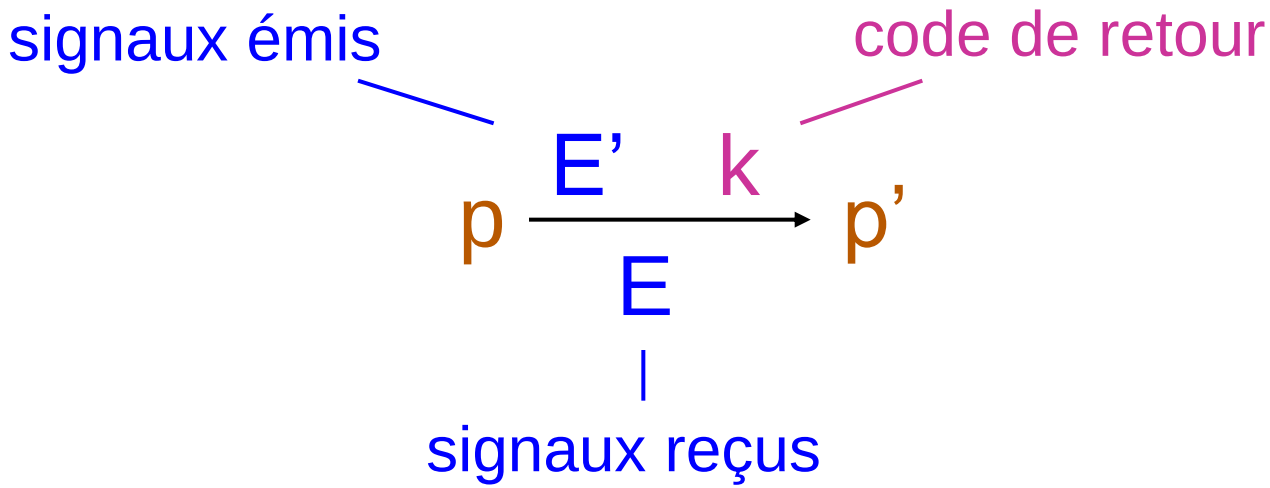
Agenda

1. Circuits digitaux
2. Des automates aux circuits
3. Esterel et SyncCharts
4. Sémantique
5. Traduction en circuits
6. Optimisation
7. Circuits cycliques

Le noyau Esterel pur

nothing	0
pause	1
emit S	! s
if S then p else q end	s ? p, q
suspend p when S	s ⊃ p
p ; q	p ; q
loop p end	p*
p || q	p | q
trap T in p end	{p} ↑ p
exit T	k, k > 1
signal S in p end	p \ s

La sémantique comportementale



Broadcasting : $E' \subset E$

k

- 0 : terminaison
- 1 : attente
- 2 : sortie du trap le plus proche
- 3 : sortie du second trap le plus proche

Numérotation des traps

```
trap T in
   trap U in
      nothing0
   ||
      pause1
   ||
      exit U2
   ||
      exit T3
   end trap
||
   exit T2
end trap
```

Si deux traps sont levés en même temps, seul le plus extérieur compte

Code de retour du parallèle
= max des codes des branches
(code de Gonthier)

$$k \xrightarrow[E]{\emptyset \quad k} 0$$

(pour $k=0$, $k=1$, $k>1$)

$$!s \xrightarrow[E]{\{s\} \quad 0} 0$$

$$\frac{s \in E \qquad p \xrightarrow[E]{E' \;\; k} p'}{s \,?\, p, q \xrightarrow[E]{E' \;\; k} p'}$$

$$\frac{s \notin E \qquad q \xrightarrow[E]{F' \;\; l} q'}{s \,?\, p, q \xrightarrow[E]{F' \;\; l} q'}$$

$$\frac{p \xrightarrow[E]{E' \quad 0} p'}{s \supset p \xrightarrow[E]{E' \quad 0} 0}$$

$$\frac{p \xrightarrow[E]{E' \quad k} p' \qquad k \neq 0}{s \supset p \xrightarrow[E]{E' \quad k} s \supset p'}$$

avec $s \supset p' = \{(s \; ? \; 1 \; , \; 2)^* \} \; ; \; s \supset p'$

$$\frac{p \xrightarrow[E]{E' \;\; k} p' \qquad k \neq 0}{p\,;\,q \xrightarrow[E]{E' \;\; k} p'\,;\,q}$$

$$\frac{p \xrightarrow[E]{E' \;\; 0} p' \qquad q \xrightarrow[E]{F' \;\; l} q'}{p\,;\,q \xrightarrow[E]{E' \cup F' \;\; l} q'}$$

$$\frac{p \xrightarrow[E]{E'\ k} p' \qquad k \neq 0}{p^* \xrightarrow[E]{E'\ k} p' ; p^*}$$

$$\frac{p \xrightarrow[E]{E'\ k} p' \qquad q \xrightarrow[E]{F'\ l} q'}{p \mid q \xrightarrow[E]{E' \cup F'\ \max(k,l)} p' \mid q'}$$

$$\frac{p \xrightarrow[E]{E' \quad k} p' \qquad k = 0 \text{ ou } k = 2}{\{p\} \xrightarrow[E]{E' \quad 0} 0}$$

$$\frac{p \xrightarrow[E]{E' \quad k} p' \qquad k = 1 \text{ ou } k > 2}{\{p\} \xrightarrow[E]{E' \quad \downarrow k} \{p'\}}$$

$\downarrow k = 1$ si $k=1$,

$= k-1$ si $k>2$

$$\frac{p \xrightarrow[E \cup \{s\}]{E' \cup \{s\}\ \ k} p'}{p \setminus s \xrightarrow[E]{E'\ \ k} p' \setminus s}$$

$$\frac{s \notin E \quad s \notin E' \quad p \xrightarrow[E]{E'\ \ k} p'}{p \setminus s \xrightarrow[E]{E'\ \ k} p' \setminus s}$$

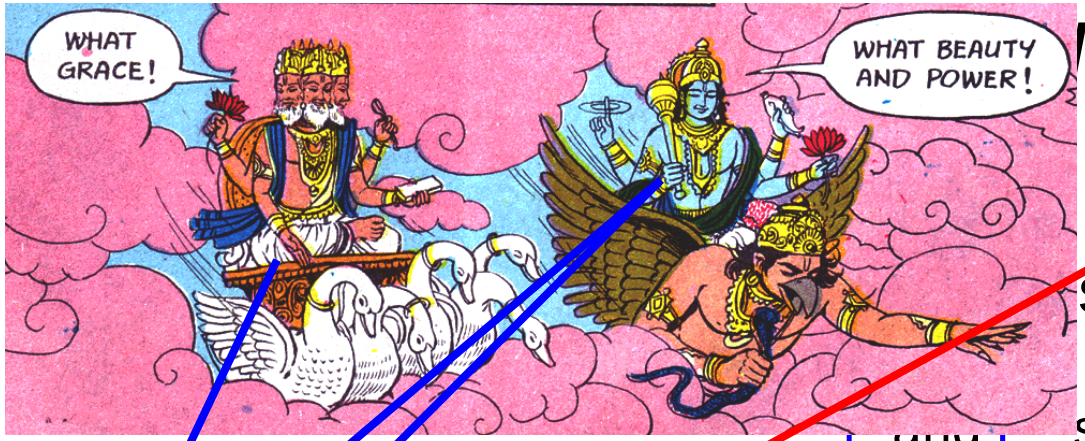
si une seule règle s'applique => déterminisme

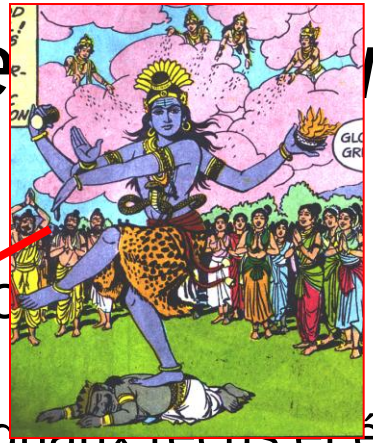
Mais les deux peuvent s'appliquer, voir plus loin!

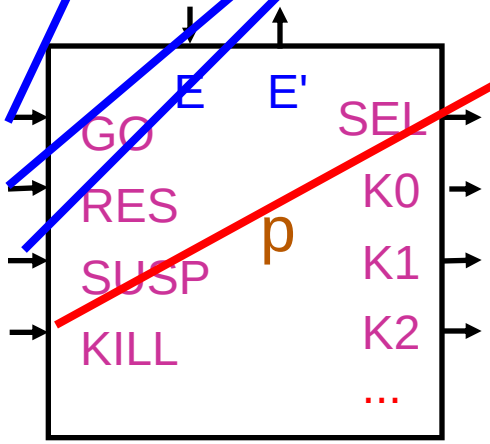
Agenda

1. Circuits digitaux
2. Des automates aux circuits
3. Esterel et SyncCharts
4. Sémantique
5. Traduction en circuits
6. Optimisation
7. Circuits cycliques

le its

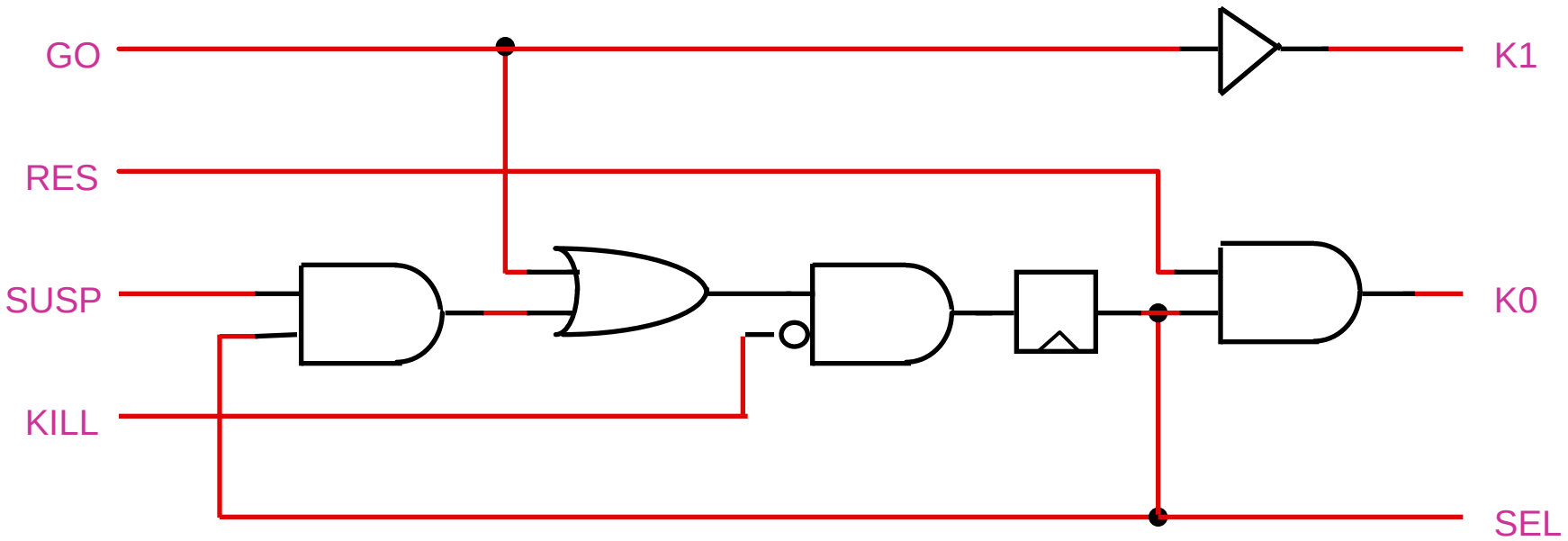




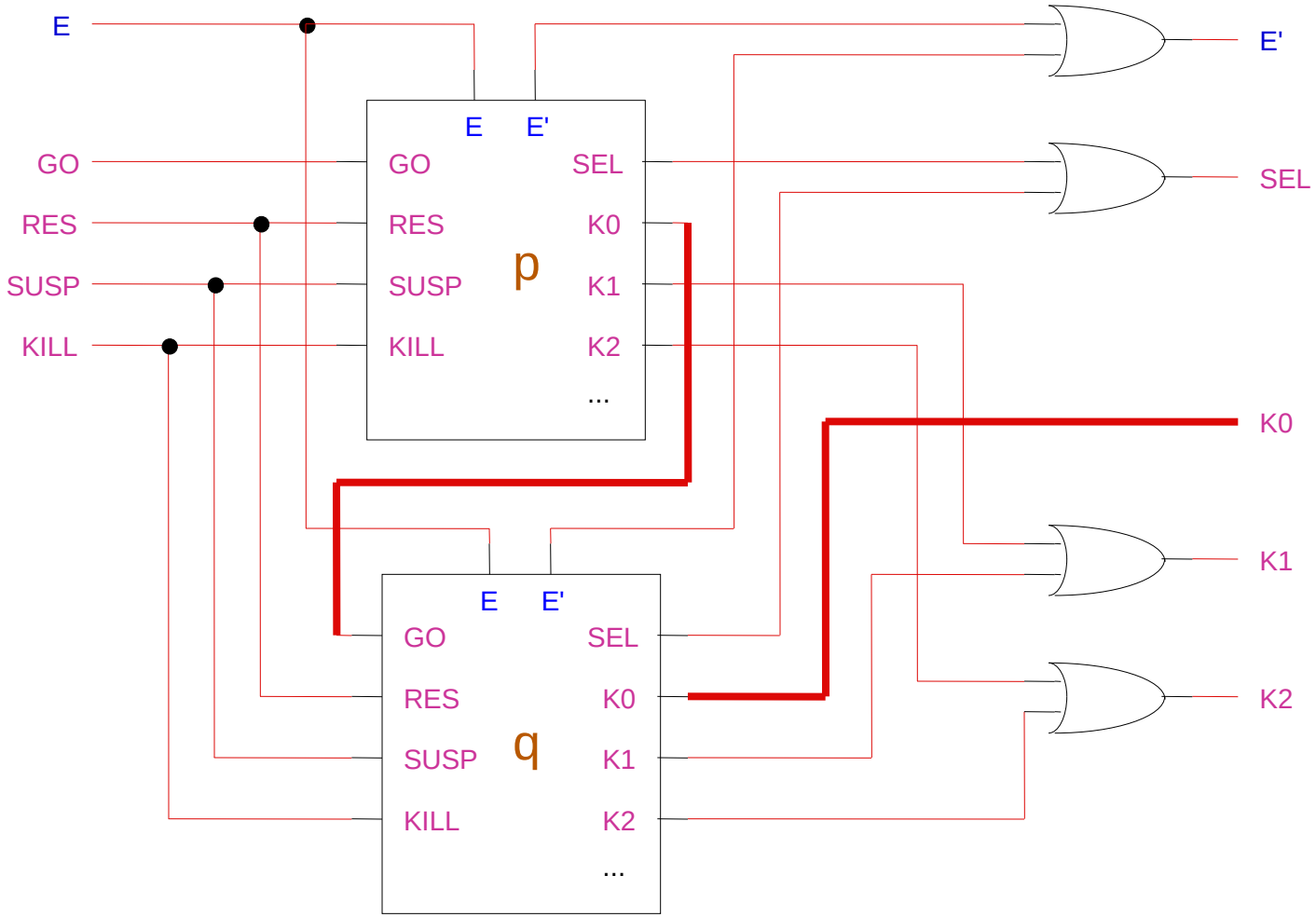


- E and E' : signaux reçus et émis
- GO : démarrer p
- RES : continuer p
- SUSP : suspendre p pour un cycle
- KILL : remettre les registres de p à 0
- SEL : au moins un registre à 1 => vivant
- Ki : *code de retour* 1-hot
  - K0: terminé
  - K1: pause, en attente d'événements
  - K2,K3,… - sortie des traps englobants

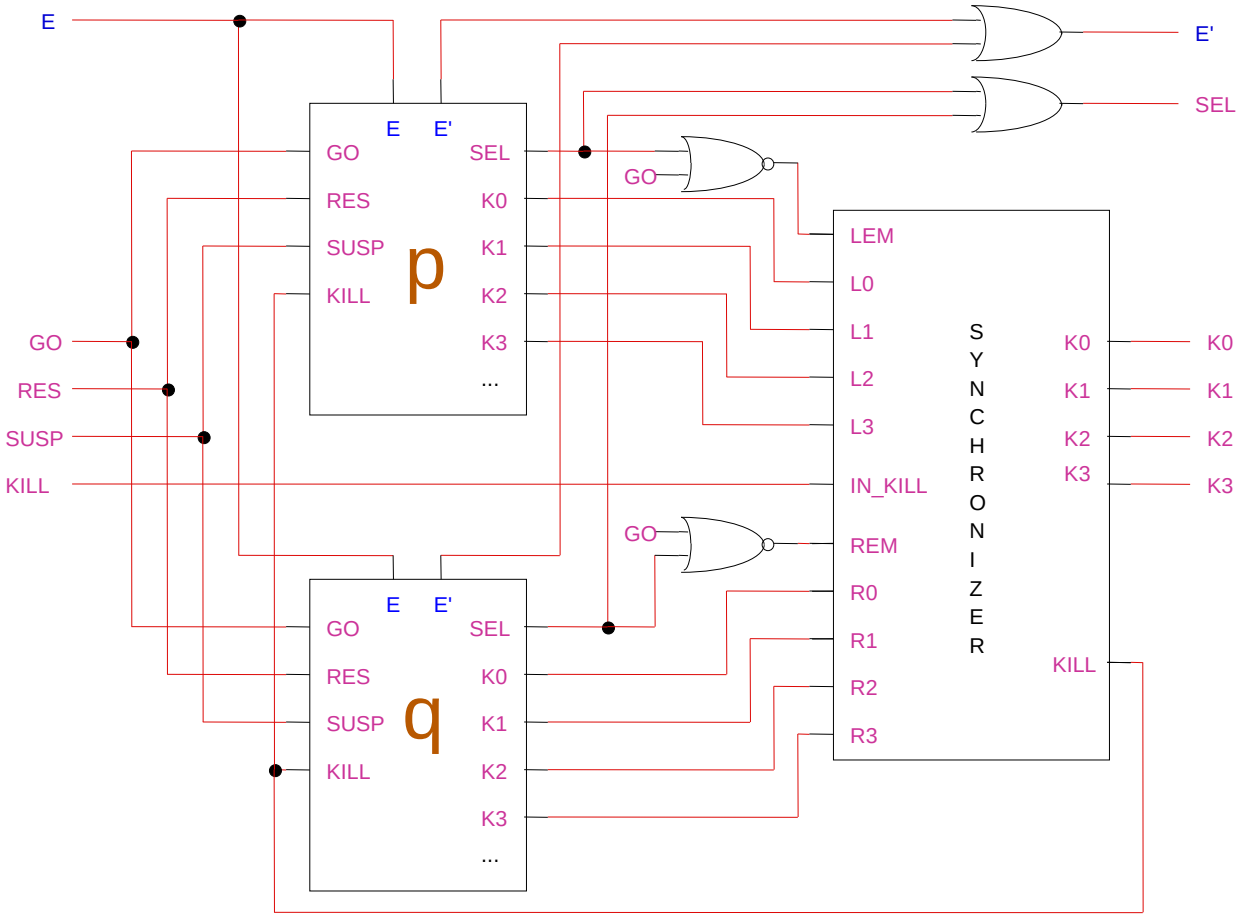
Circuit pour 1 (pause)



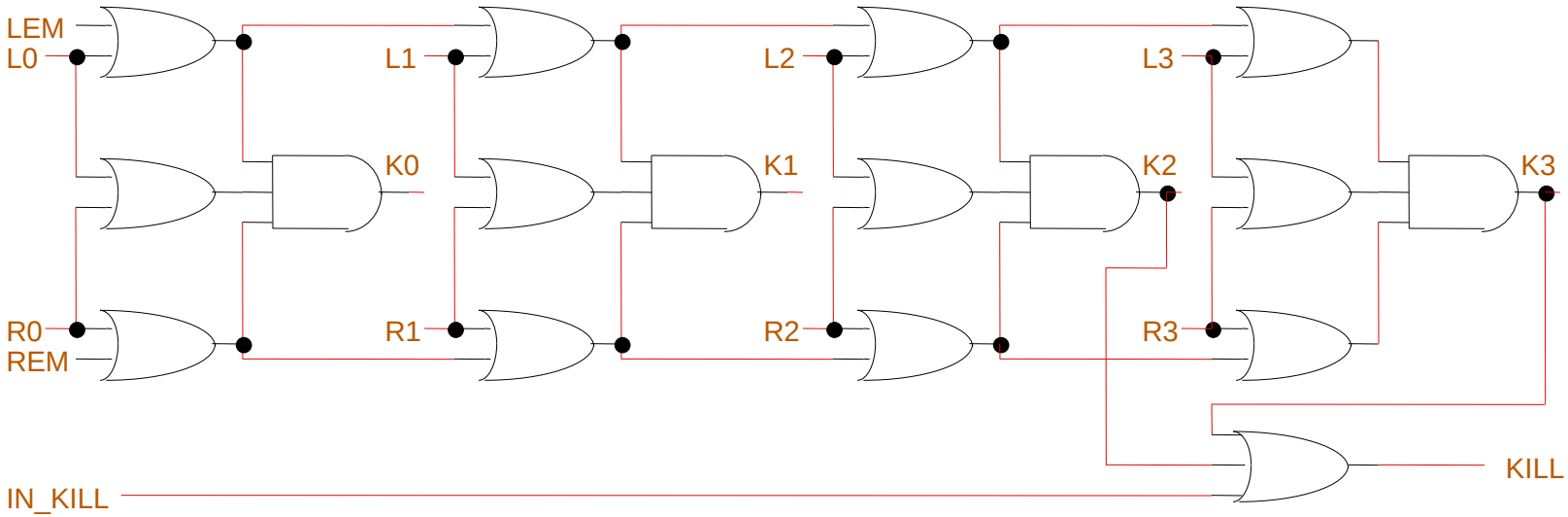
Circuit pour « p ; q »



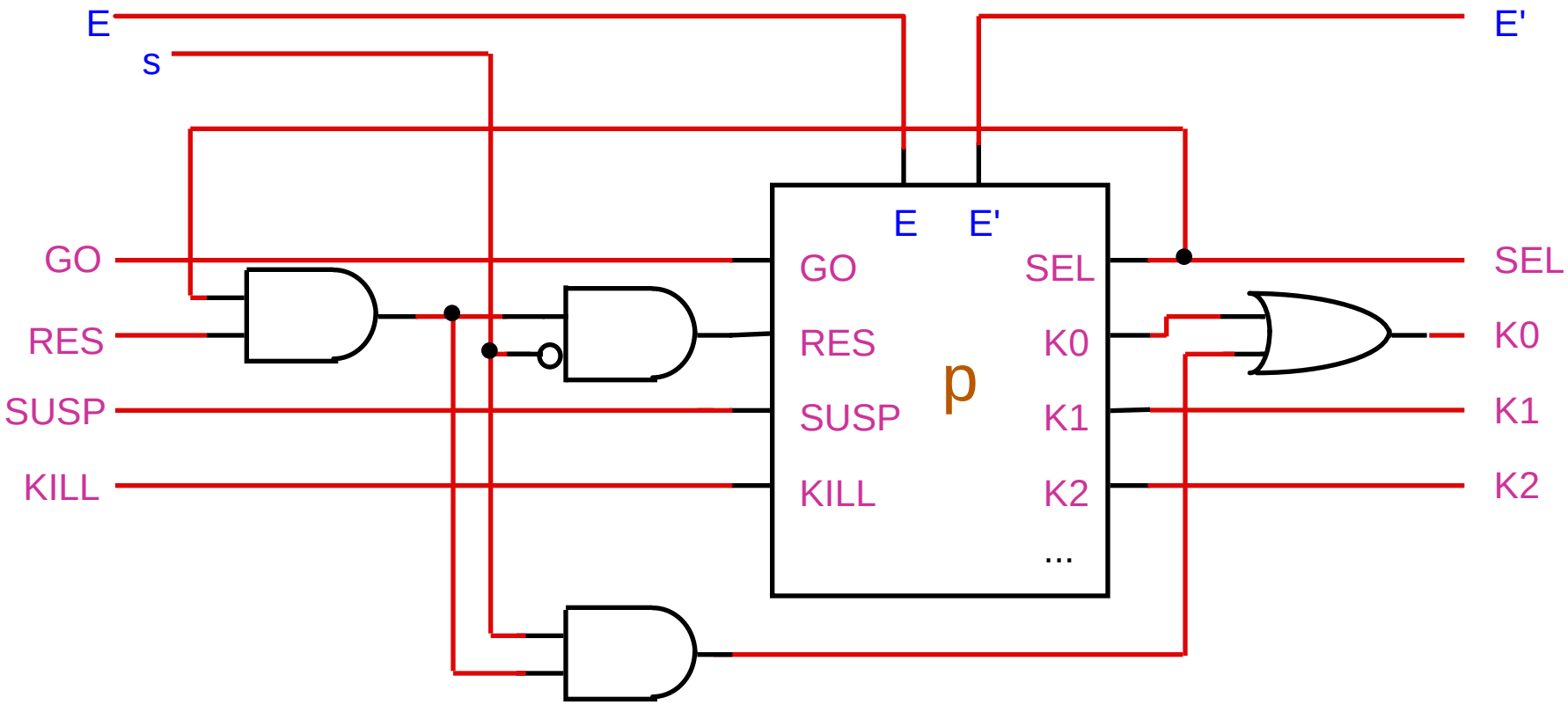
Circuit pour « p || q »



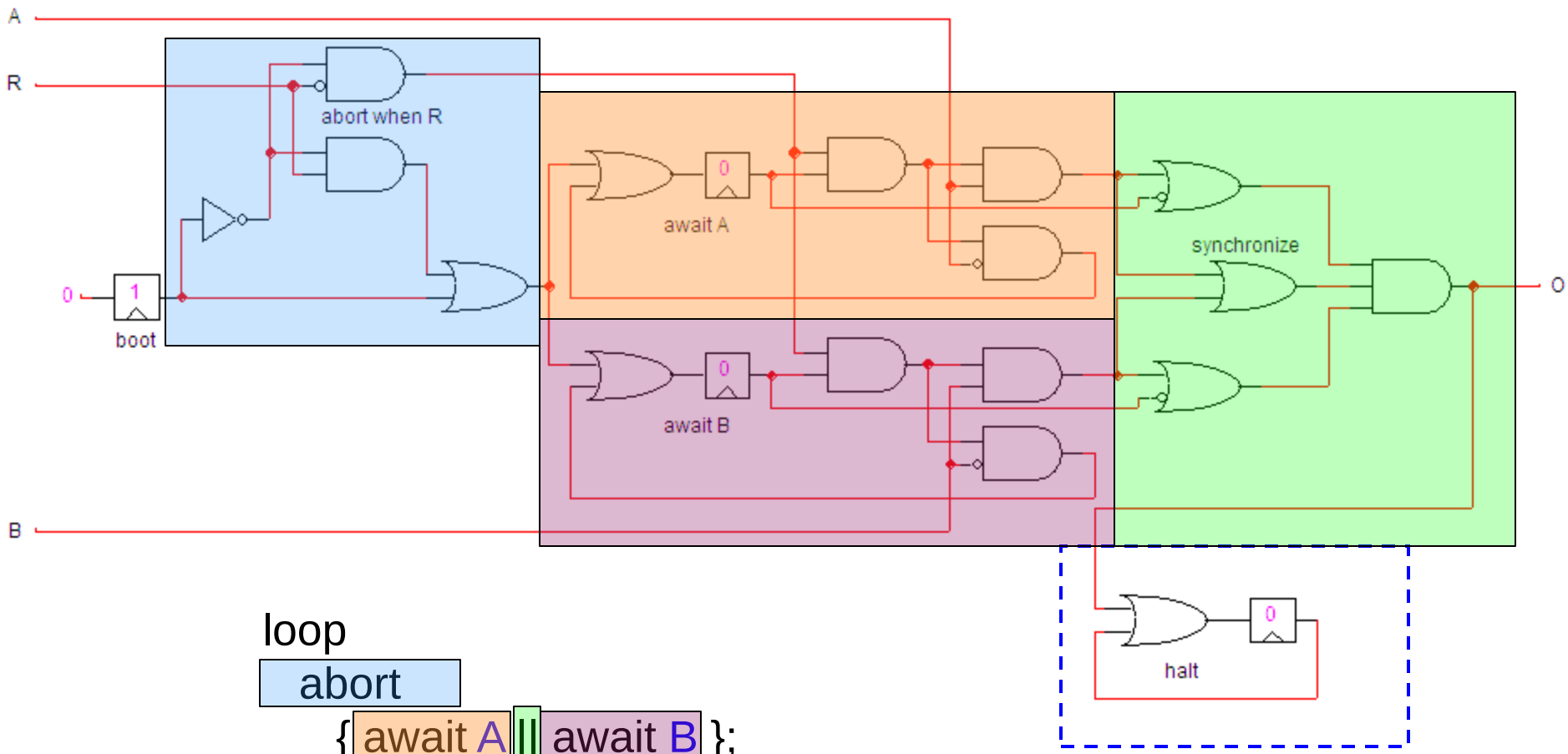
Le synchroniseur du parallèle



Circuit pour « abort p when s »



Le circuit ABRO hiérarchique




```
loop
  abort
      { await A || await B };
      emit O ;
      halt
  when R
end loop
```

saute à l'optimisation

Agenda

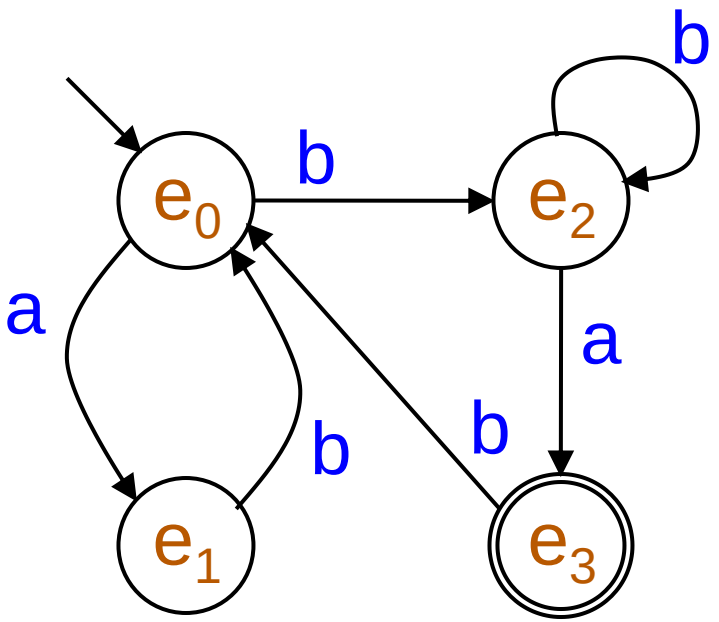
1. Circuits digitaux
2. Des automates aux circuits
3. Esterel et SyncCharts
4. Sémantique
5. Traduction en circuits
6. Optimisation
7. Circuits cycliques

Optimisation séquentielle

1. Construction du circuit hiérarchique
   bon début, mais trop gros
2. Optimisation des registres
   réduction des redondances
3. Optimisation de la logique combinatoire
   vitesse, surface, puissance dissipée

Calcul Booléen à grande échelle
(expressions, BDDs, SAT, cf cours algo. 2007-2008)

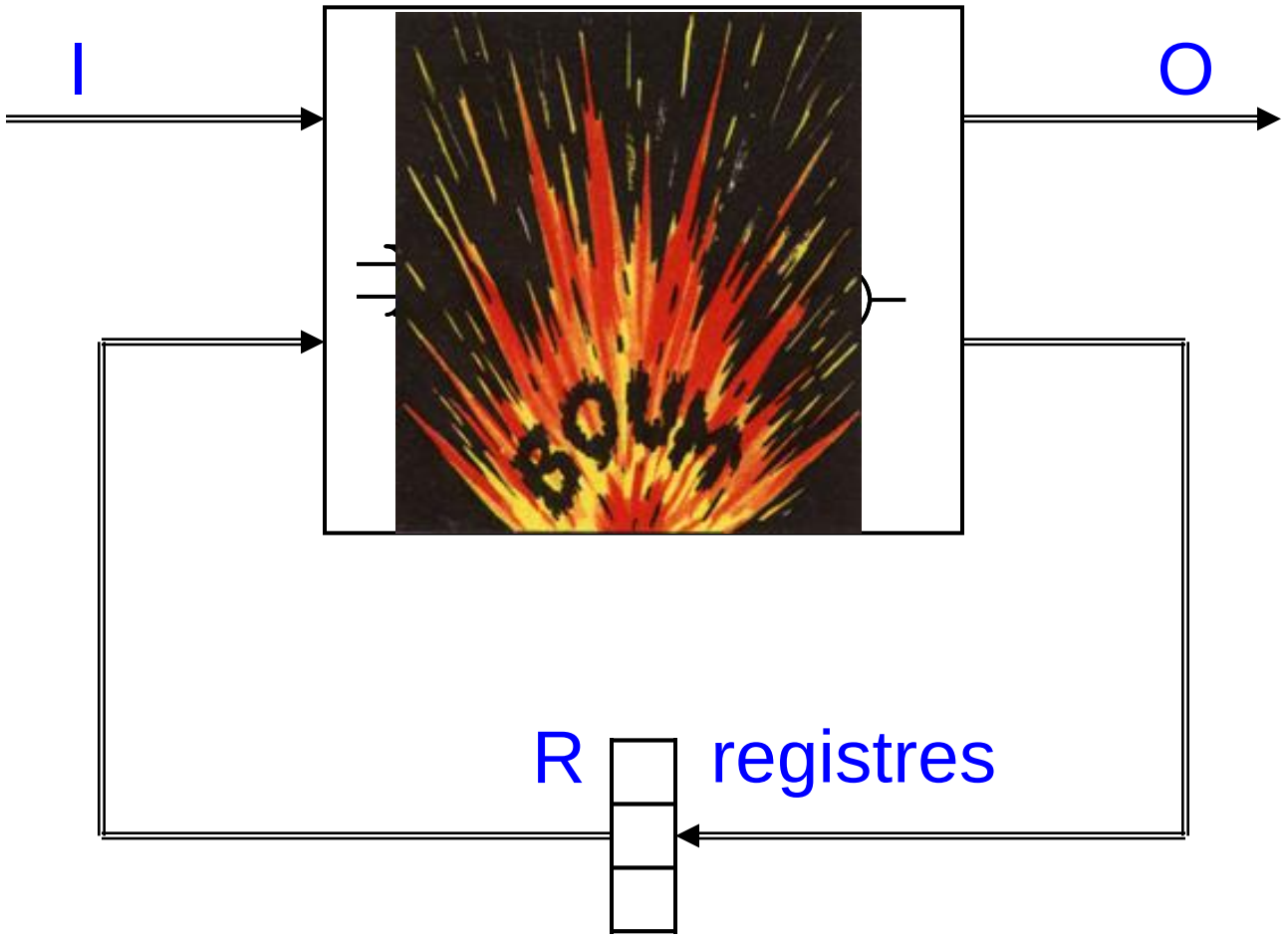
Mauvaise solutions



- déterministe => 1-hot
  explose avec le nb d'états
- log(n) bits pour n états
  peut faire exploser la logique

$e_0 = 10$	$e_0 = 01$
$e_1 = 11$	$e_1 = 10$
$e_2 = 01$	$e_2 = 11$
$e_3 = 00$	$e_3 = 11$
bon	mauvais

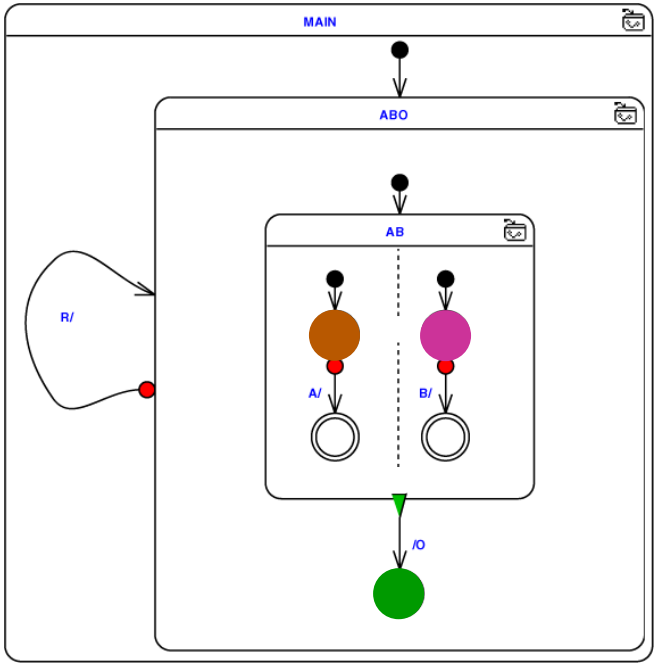
n! possibilités, pas d'heuristique !



Il faut équilibrer les registres et la logique
Solutions : automates non-déterministes
encodage structurel d'Esterel / SyncCharts

Le secret : écrire chaque chose une fois !

```
loop
   abort
      { await A || await B };
       emit O ;
      halt
   when R
end loop
```

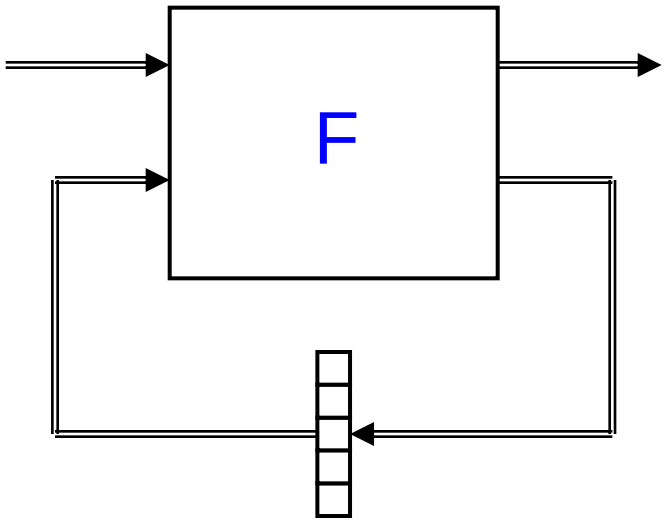


Un registre par attente explicite

$\Rightarrow$ bon équilibre registres / logiques

Plus le programme est bien écrit, plus il est efficace !

Vérification, optimisation : calculer les états atteignables



$A_0 = 0$

$A_1 = A_0 \cup F(A_0)$

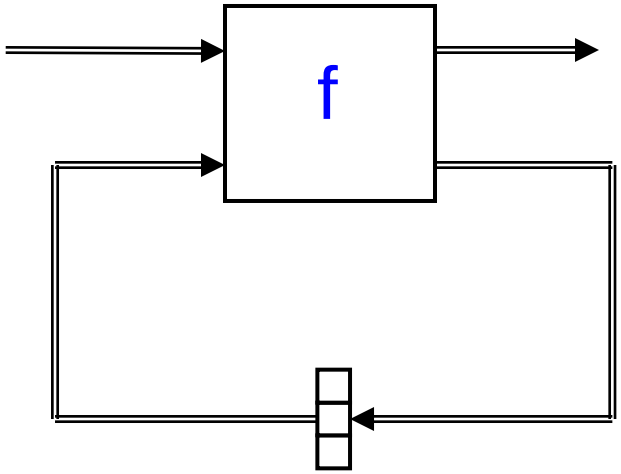
$A_2 = A_1 \cup F(A_1)$

...

$A = \cup A_i$

- Calculer les A_i en utilisant des BDDs - mais .... BDD(F) explose !
- Utiliser chaque A_i comme simplifieur pour BDD(F)
- Puis utiliser A comme optimiseur pour l'implémentation de F (Madre, Coudert, Touati)

Vérification, optimisation : calculer les états atteignables

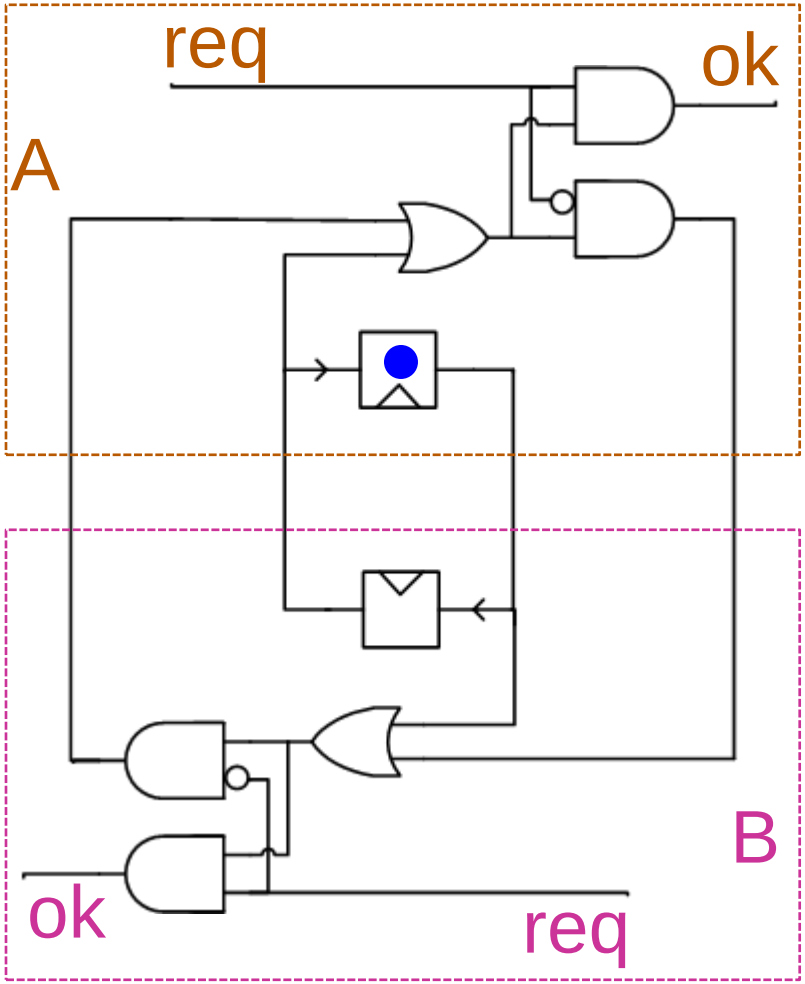


Esterel v7 : en pratique, toujours meilleur que les méthodes traditionnelles

Agenda

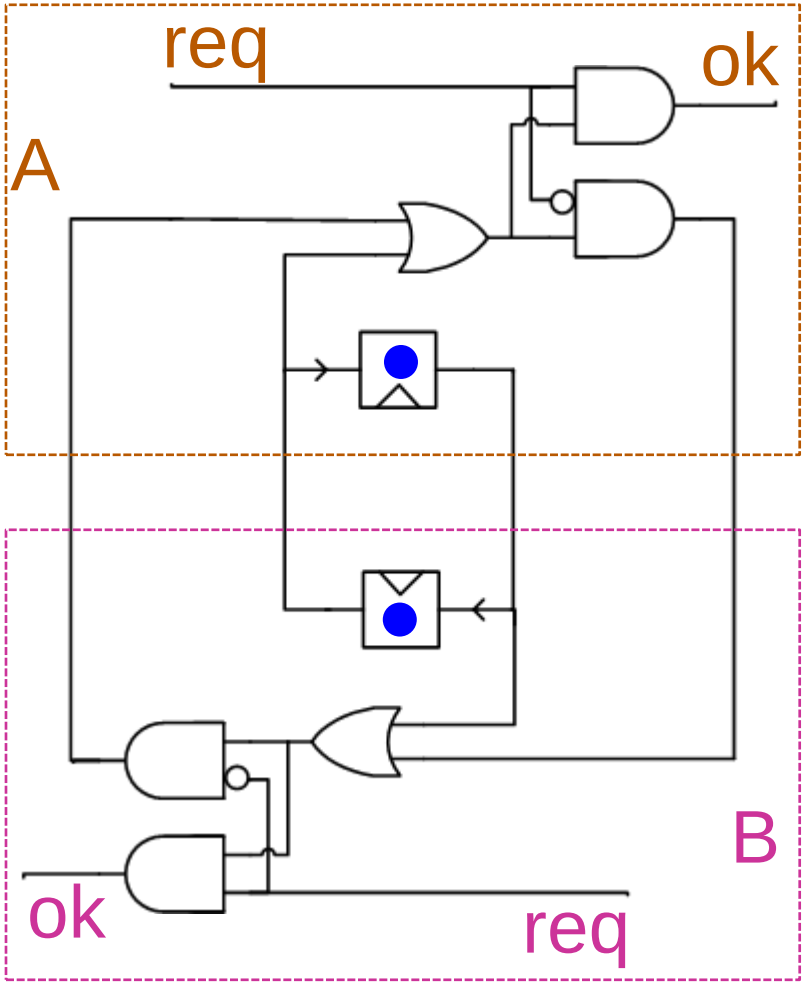
1. Circuits digitaux
2. Des automates aux circuits
3. Esterel et SyncCharts
4. Sémantique
5. Traduction en circuits
6. Optimisation
7. Circuits cycliques

Protocole round-robin cylique à 2



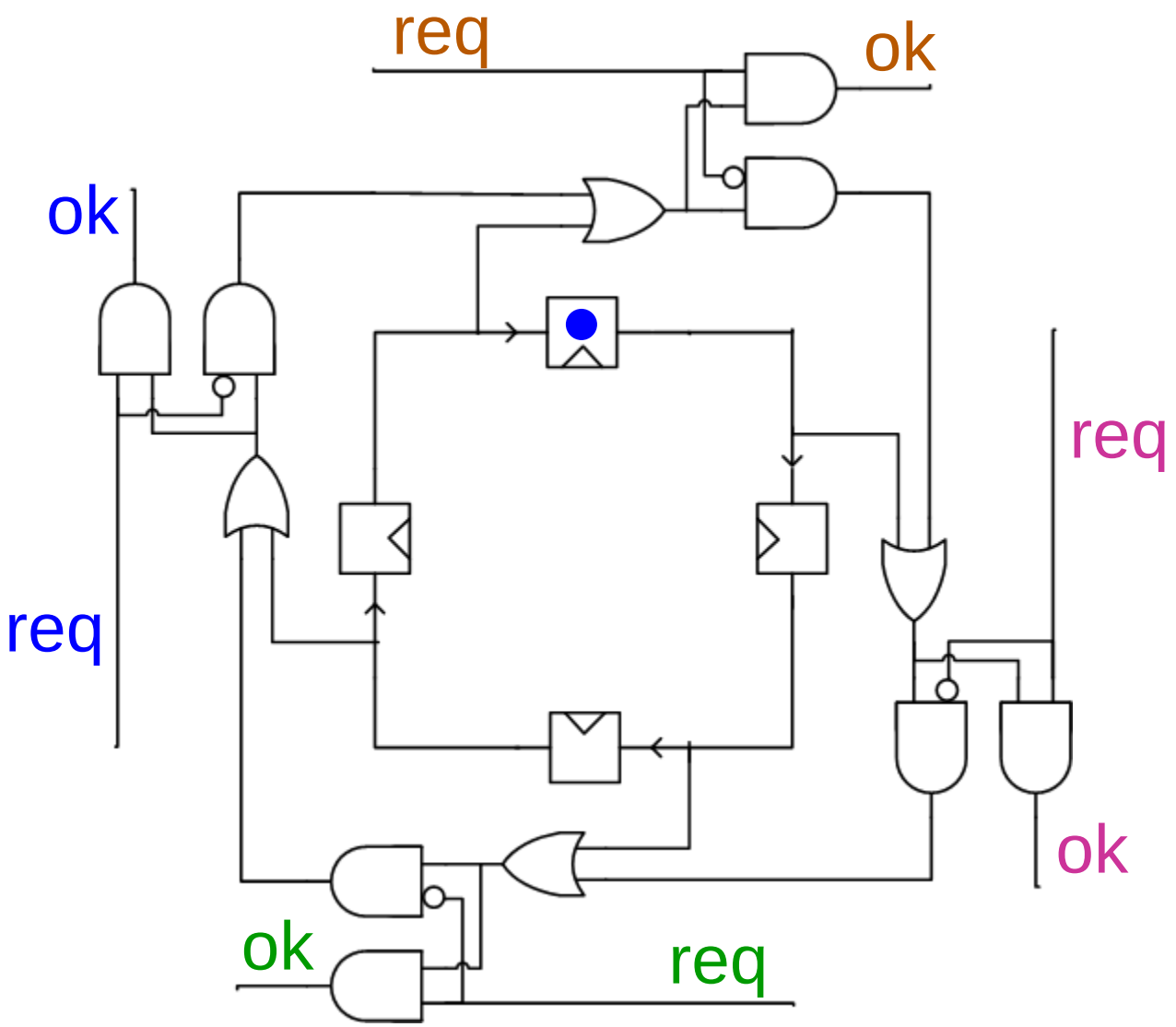
- un seule requête ok dans l'ordre des demandes après le registre à 1

Protocole round-robin cylique à 2

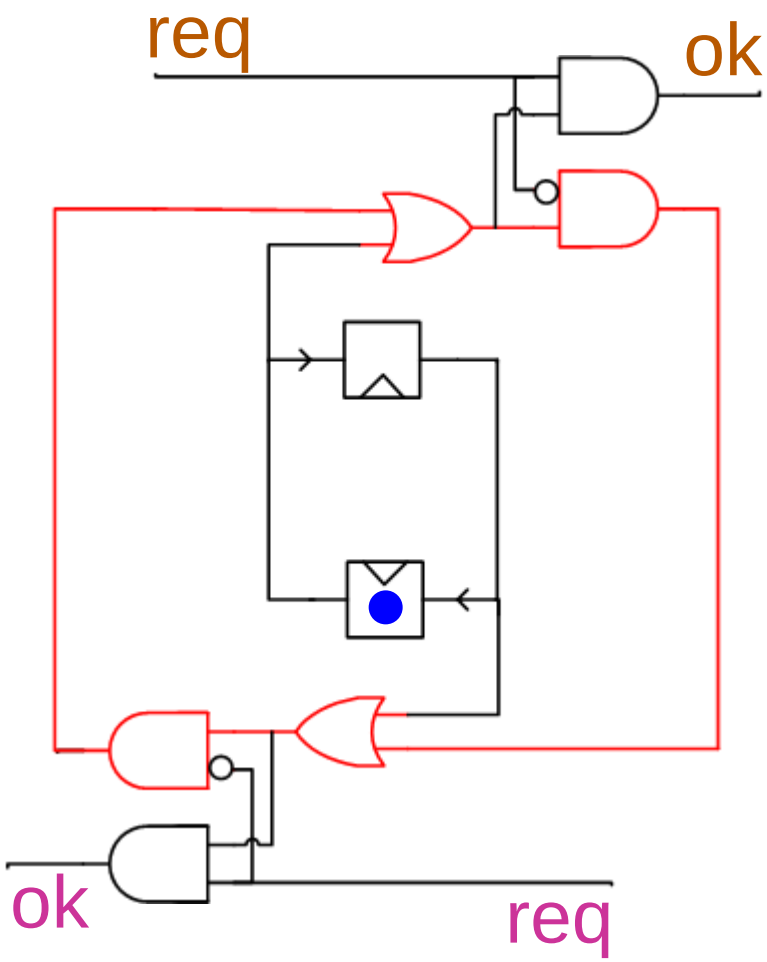


- un seule requête ok dans l'ordre des demandes après le registre à 1
- qui tourne à chaque fois

Protocole round-robin cylique à 4



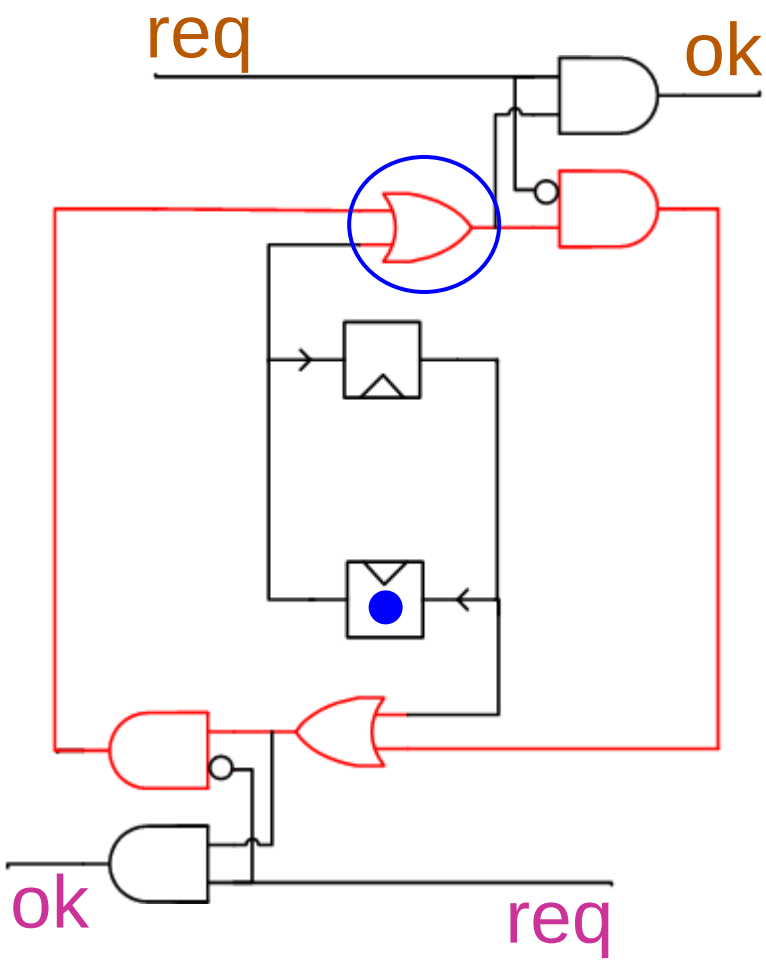
Protocole round-robin cylique à 2



- un seule requête ok dans l'ordre des demandes après le registre à 1
- qui tourne à chaque fois

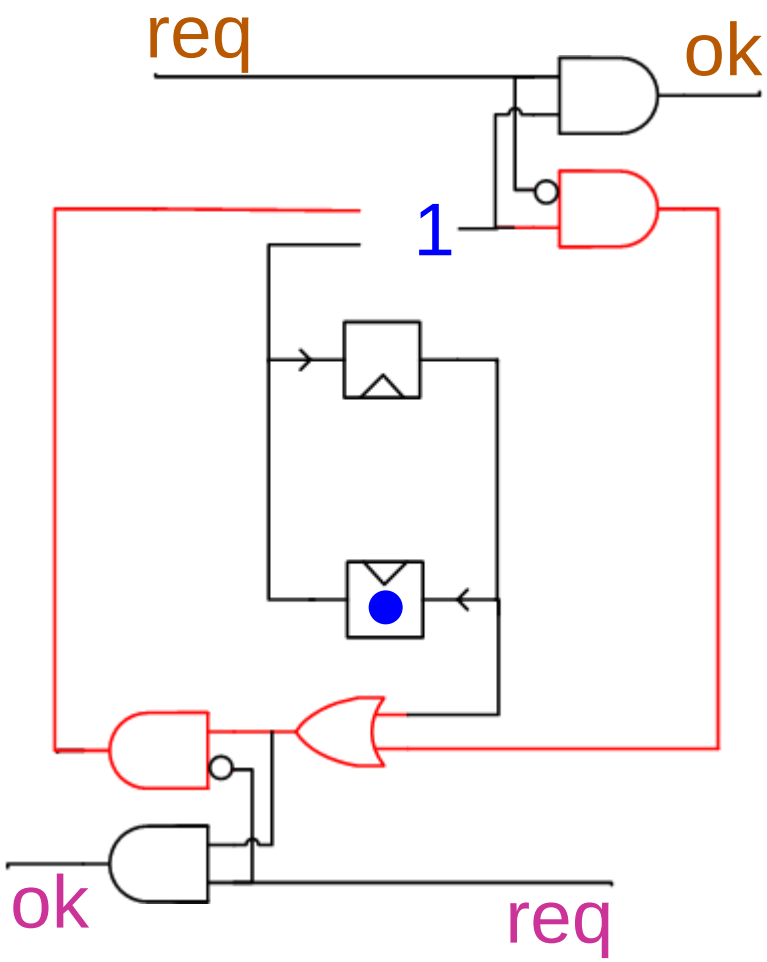
Attention !
cycle combinatoire !

Protocole round-robin cylique à 2



Cycle sain ssi
au moins un registre vaut 1
car coupé par tout registre à 1

Protocole round-robin cylique à 2



Cycle sain ssi
au moins un registre vaut 1
car coupé par tout registre à 1

Trois sortes de circuits cycliques

1. Sans espoir, ni électriquement, ni logiquement

   X = X

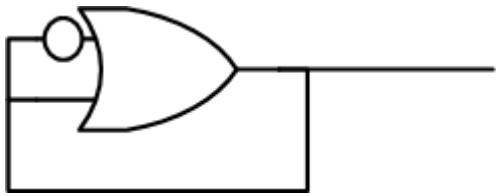
   X = non X

2. Sans problème (éventuellement sous conditions)

   partie combinatoire du round-robin cyclique
   (si au moins une sortie de registre à 1)

3. Etrange

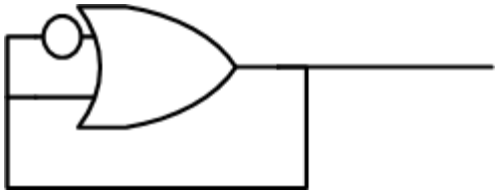
   Hamlet : ToBe = ToBe or not ToBe



Trois sortes de circuits cycliques

3. Etrange

Hamlet : ToBe = ToBe or not ToBe



Se stabilise électriquement à 1 pour certains délais des portes et fils, mais pas pour tout les délais

Quand un circuit se stabilise-t-il
pour tous les délais des portes et des fils ?

L'électricité est constructive!
(Berry – Shiple – Mendler)

Théorème : pour une entrée donnée un circuit se stabilise pour tous délais des portes et des fils

- si et seulement si toutes les valeurs de ses fils sont calculables en logique constructive, sans tiers exclu « X ou non X = 1 »
- si et seulement si le point fixe de ses équations dans la logique de Scott ($\mathbb{B}_\perp$) est partout défini

$\Rightarrow$ il existe des délais pour lesquels Hamlet ne se stabilise pas !

Logique propositionnelle constructive

- E : environnement = fonction des entrées I dans {0,1}
- Formules : « E prouve X = b », écrit « $E \vdash X = b$ »

$$\frac{I \text{ entrée}}{E \vdash I = E(I)} \qquad \frac{E \vdash e = 0}{E \vdash \text{non } e = 1} \qquad \frac{E \vdash e = 1}{E \vdash \text{non } e = 0}$$

$$\frac{E \vdash e = 0}{E \vdash e \text{ et } e' = 0} \qquad \frac{E \vdash e' = 0}{E \vdash e \text{ et } e' = 0} \qquad \frac{E \vdash e = 1 \quad E \vdash e' = 1}{E \vdash e \text{ et } e' = 1}$$

$$\frac{E \vdash e = 1}{E \vdash e \text{ ou } e' = 1} \qquad \frac{E \vdash e' = 1}{E \vdash e \text{ ou } e' = 1} \qquad \frac{E \vdash e = 0 \quad E \vdash e' = 0}{E \vdash e \text{ ou } e' = 0}$$

$$\frac{X = e \quad E \vdash e = b}{E \vdash X = b}$$

Conclusion

- Le modèle synchrone déterministe
  - parallélisme massif + déterminisme
  - avec toutes les bonnes propriétés du séquentiel
  - fondé sur des sémantiques mathématiques rigoureuses
  - qui conduisent à d'excellentes propriétés d'implémentation

- Son champ d'application
  - reste limité à des systèmes relativement compact
  - mais reste considérable :
    circuits, systèmes embarqués critiques (avions, trains, etc.)

le 27 janvier :
marier le séquentiel, le synchrone et l'asynchrone

Références

- *Synchronous Programming of Reactive Systems*
  Nicolas Halbwachs
  Kluver, 1992

- *Statecharts : a Visual Formalism for Complex Systems*
  David Harel
  Science of Computer Programming 8 (1987) 231-274

- *The Foundations of Esterel*
  Gérard Berry
  dans “Proof, Language and Interaction: Essays in Honour of Robin Milner, MIT Press, Foundations of ComputingSeries, 2000.

Références

- *Compiling Esterel*
  Dumitru Potop-Butucaru, Stephen Edwards et Gérard Berry
  Springer, 2008

- *The Constructive Semantics of Pure Esterel*
  Gérard Berry, web book

- ...et plus sur www-sop.inria.fr/members/Gerard/Berry