NoC - DSPIN

SMC - MU5IN162

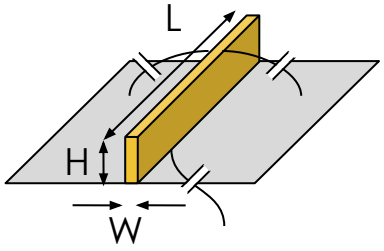
NoC - Objectifs

- Besoins
  - 1 core - 1 mémoire : pas de problèmes
    → La mémoire peut répondre à toutes les requêtes
  - N cores - 1 mémoire
    → Si N est inférieur à ≤ 10, la mémoire peut répondre à toutes les requêtes si les cores disposent de caches performants (write back et larges), en write-through N ≤ 4
  - N cores - M mémoire
    → On peut adapter le nombre de mémoires au nombre de cores, pour que la mémoire puisse répondre ∀ N mais le problème, c'est l'interconnect entre les cores et la mémoire
- Les BUS ne sont pas une réponse

Micro-réseau

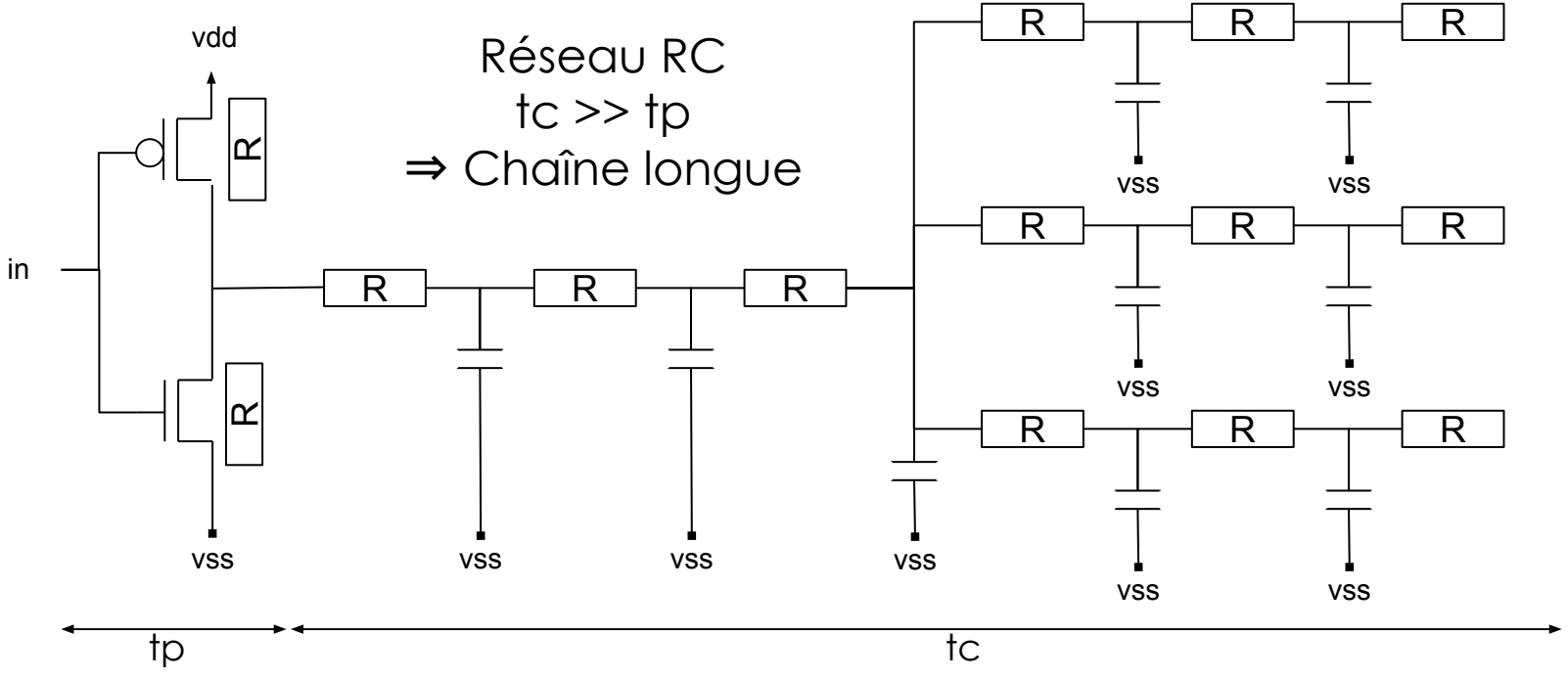
But	Plusieurs transactions simultanées
Problèmes	Risque d'interblocage dans le réseau
	Il faut éviter les fils longs

Problème des fils longs

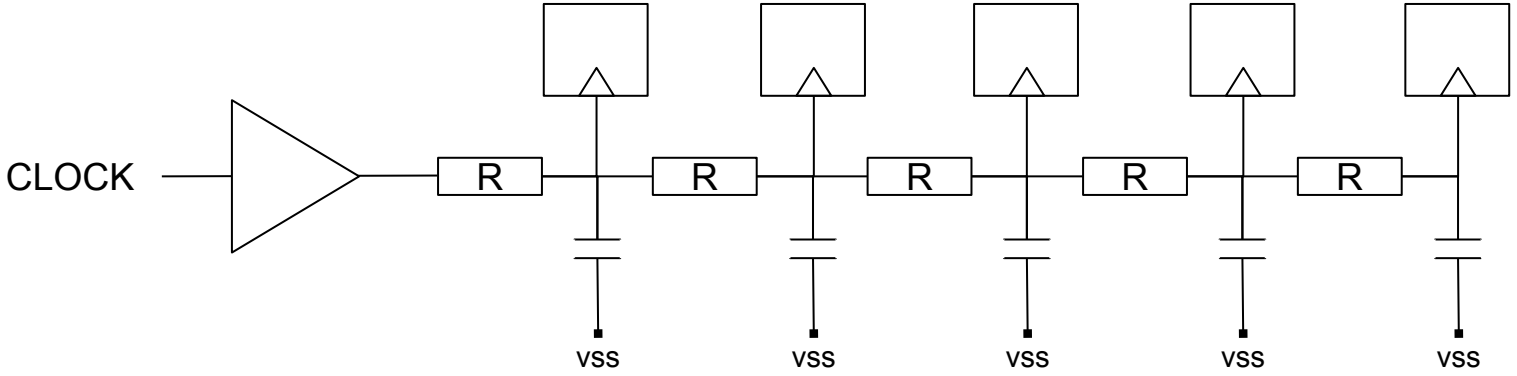


Avec l'évolution des procédés de fabrication, la section (WH) diminue mais pas forcément la longueur, OR la résistance R = *k* L/(WH)

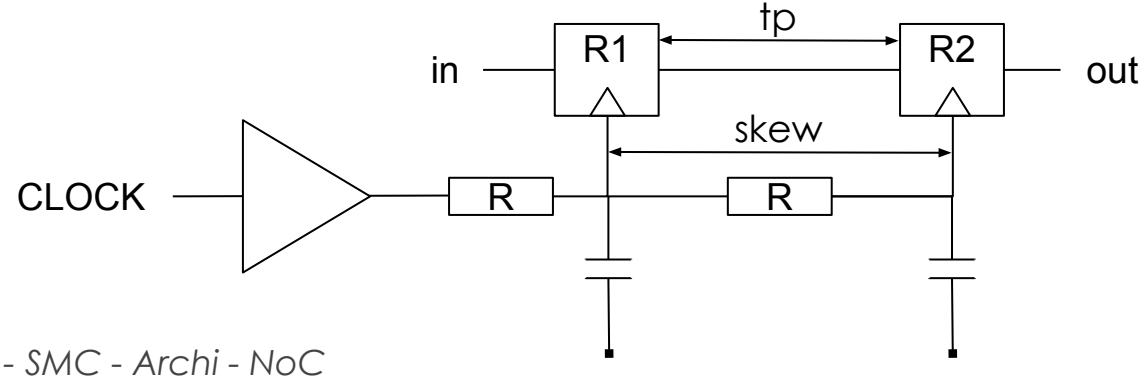
Réseau RC
tc >> tp
⇒ Chaîne longue



Problème de distribution d'horloge

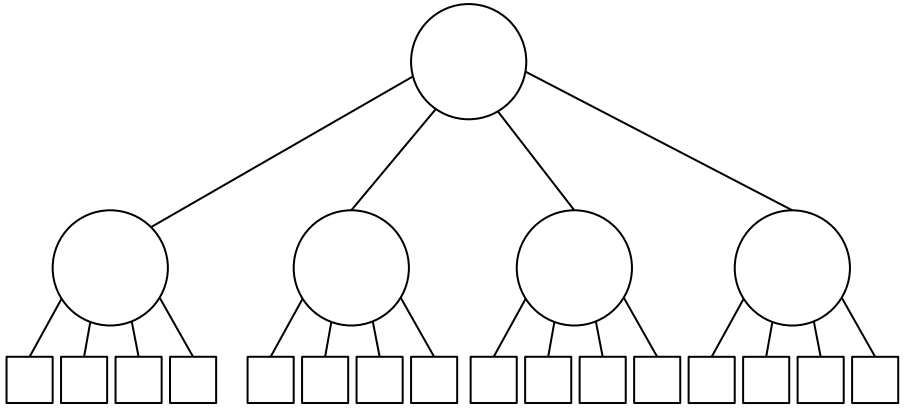


Le signal d'horloge n'est plus synchrone, il y a un décalage temporel entre les bascules (skew) qui peut créer des chaînes courtes.



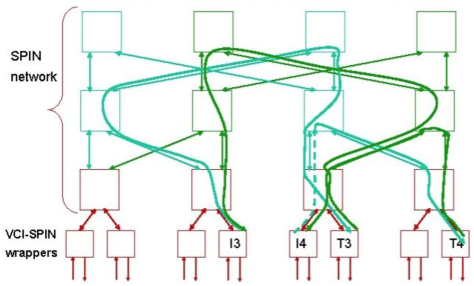
si tp < skew
alors R1 **disparait**

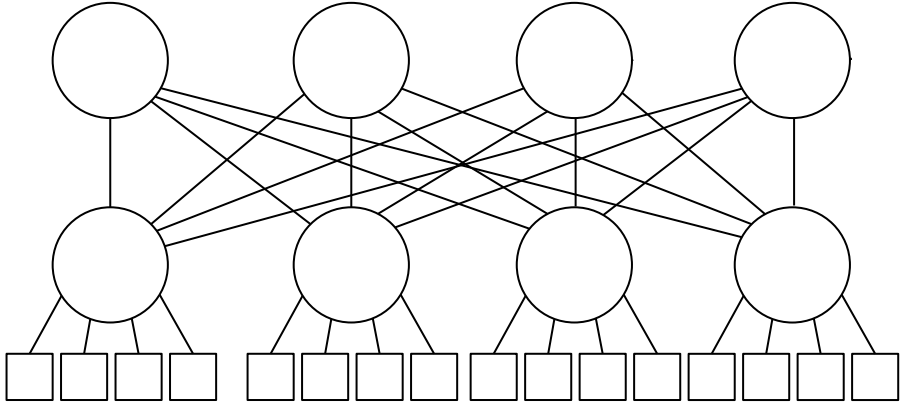
SPIN - Présentation



- Conçu au LIP6 et développé par STMicro dans les années 2000 (https://link.infini.fr/m9KbV1Jf)
- Topologie d'arbre quaternaire
- Chaque fil,
  c'est une nappe montante
  et une nappe descendante
- La racine de l'arbre voit les ¾ des commandes et des réponses échangées

SPIN - Présentation

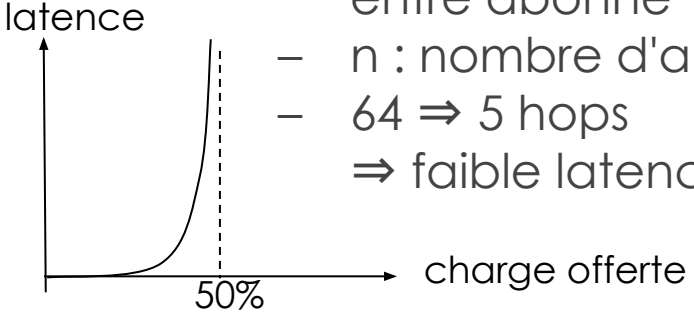




- C'est un Fat Tree (arbre élargi)
  Il en existe plein de types
- Augmente la bande passante
- Réseau adaptatif
  - plusieurs routes possibles pour réduire les contentions
  - *out-of-order delivery*
- Diamètre : D = log4(n) +1
  - plus grande distance entre abonné
  - n : nombre d'abonnés
  - 64 ⇒ 5 hops
    ⇒ faible latence

Seuil de saturation ≈ 50%

- trafic synthétique
- buffer en sortie infinie
- 1 échange tous les 2 cycles

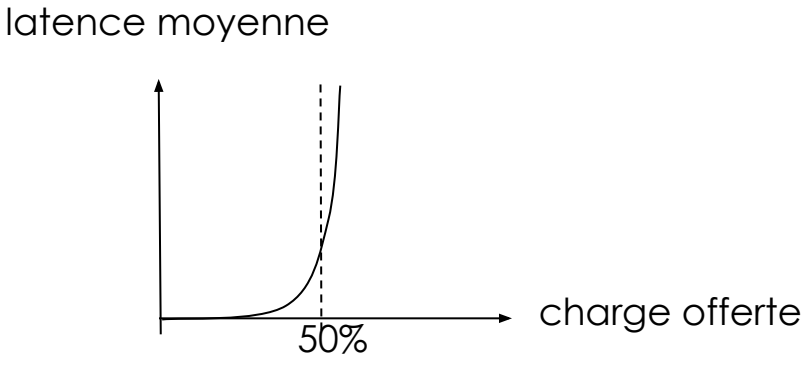


SPIN - Problème

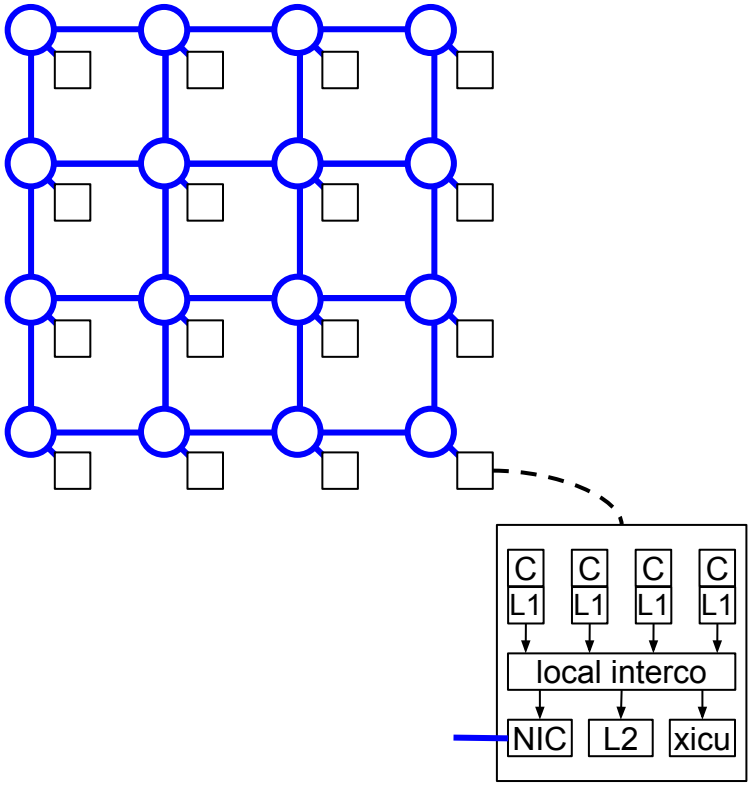
- Long fils
  - la longueur des fils dépend des routeurs
  - FIls longs ⇒ très résistif ⇒ temps de propagation majoritaire
  - Sensibilité au bruit et donc risque de corruption de donnée
  - Difficulté de routage si l'arbre est large (beaucoup d'abonnés)
- Adpatabilité
  - Plusieurs chemins entre deux noeuds ⇒ perte de consistance
  - Algorithme logiciel plus complexe (numérotation des paquets)
  - mécanisme pour savoir si un paquet est coincé ou perdu
    - timeout
    - voiture balai

Charge d'un réseau

- Ce qui caractérise les performances sont la latence et le débit
- Pour connaître la qualité d'un réseau
  - chaque source a des transactions avec des cibles uniformément distribuées.
  - La durée de la transaction va dépendre de la charge.
- soit ρ le nombre de flits offerts pour une transaction par unité de temps $\rho \in [\varepsilon,1]$
  $\rho = L / (L+Gm)$ avec
  L = longueur paquet et
  Gm = nombre moyen de cycles entre paquets
- La saturation
  c'est la fonction qui fait
  dépendre la latence de
  la charge offerte



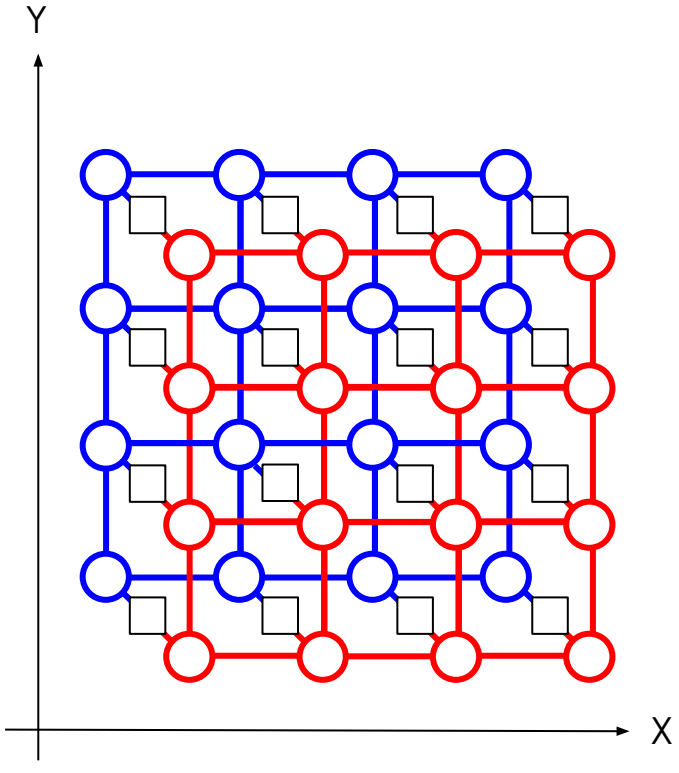
DSPIN - Présentation



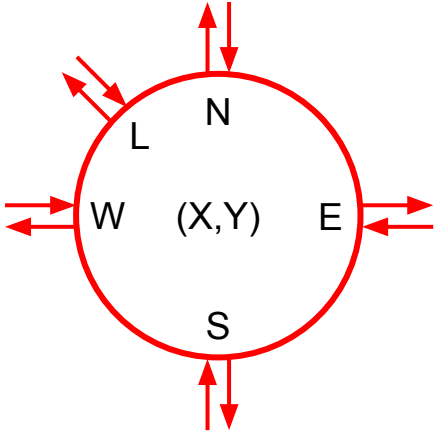
Distributed SPIN

- 2D - mesh
- plus de long fils
- politique de routage déterministe
- Diamètre : D = $2 * \sqrt{n} - 1$
  - ici D = 7 (SPIN : D = 3)
- GALS : Globally Asynchronous
  Locally Synchronous
  chaque « abonné » a son horloge
  ⇒ problème de franchissement des frontières d'horloge
- Abonné = sous-système
  - 4 cores
  - Mémoire (L2)
  - périphériques internes (ICU, timer, etc.)
  - NIC Network Interface Controler

DSPIN - Présentation



- Commandes et réponses ont leur propre 2D-mesh pour éviter les interblocages
- Les paquets sont routés de manière atomiques (comme pour VCI)



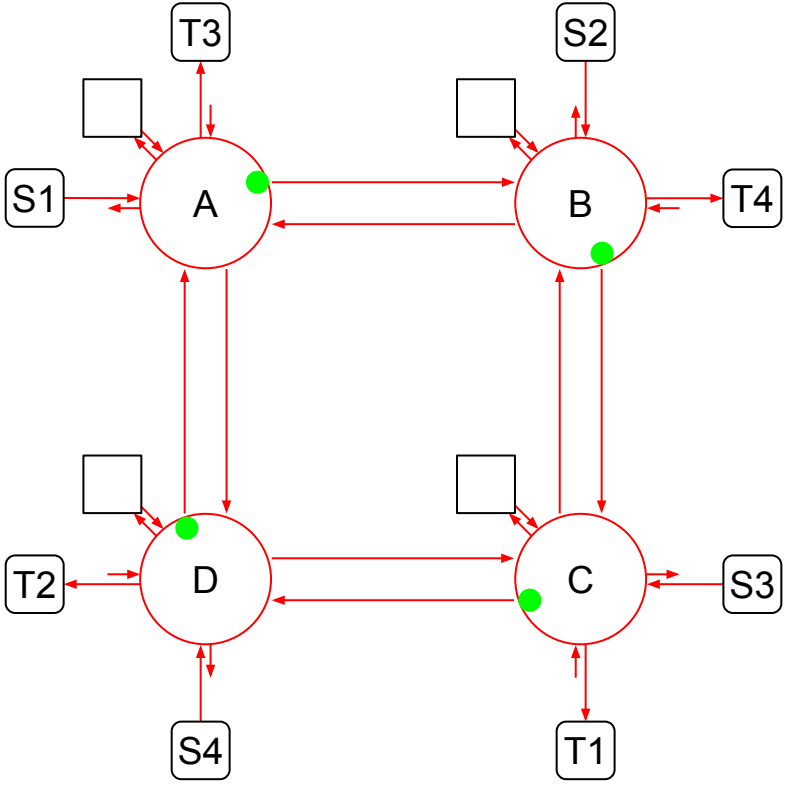
les nœuds sont

- crossbar 5 ports
  (3 ou 4 ports à la périphérie)
  1 port = entrée + sortie
- définis par leur coordonnées dans le 2D-mesh

DSPIN - algorithme de routage

- Définit le chemin à prendre entre deux nœuds de `(X,Y)` à `(X',Y')`
  - déterministe ⇒ 1 seul chemin pour garantir le *in-order-delivery*
  - deadlock free
- Chaque port d'entrée a une fonction de routage : `f(Xd,Yd,X0,Y0)`
  - `(Xd,Yd)` destination ; `(X0,Y0)` routeur local
  - choisit le port d'entrée cela permet un routage en parallèle des paquets entrants, c'est combinatoire (pas d'automate).
- Chaque port de sortie a un arbitre qui garantie l'équité (*round robin*)
- 2 techniques pour le routage des paquets
  - l'entête contient la liste des ports de sorties à prendre, c'est un routage à la source
    ⇒ entête de taille dépendant de la distance à parcourir
  - choix local dépendant d'un calcul sur les coordonnées
    ⇒ risque d'interblocage

DSPIN - Fonction de routage

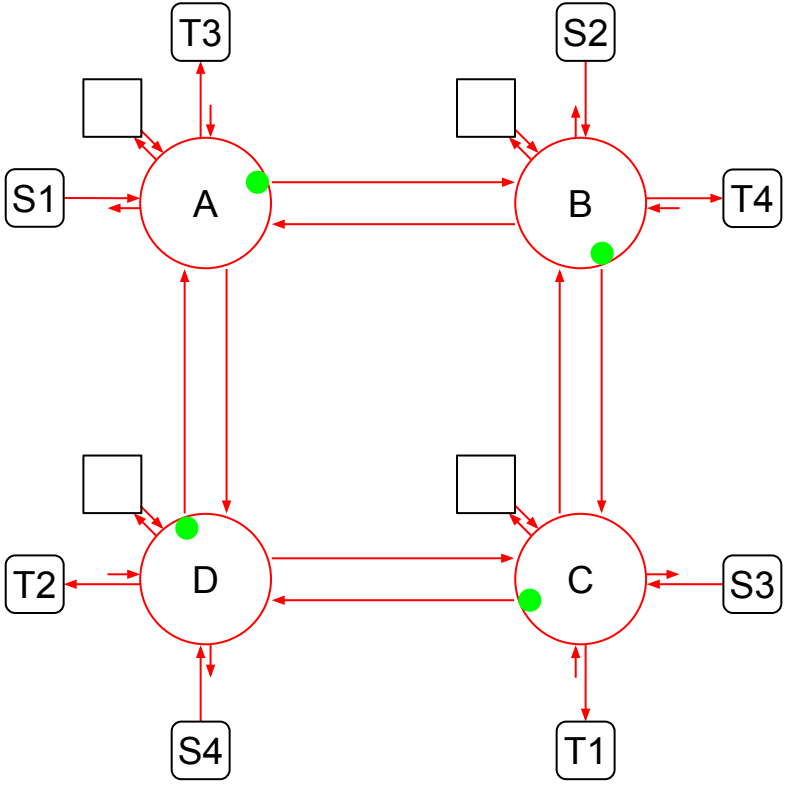


Prenons un exemple : px = Sx → Tx

- p1, p2, p3 et p4
  entrent en même temps
  ⇒ ils allouent un port de sortie ●
- p1, p2, p3 et p3
  entre dans le routeur suivant
  ⇒ ils sont tous bloqués parce qu'ils doivent allouer un port de sortie déjà alloué

⇒ c'est un deadlock
étreinte fatale c'est plus parlant

DSPIN - Analyse du deadlock



Besoins :

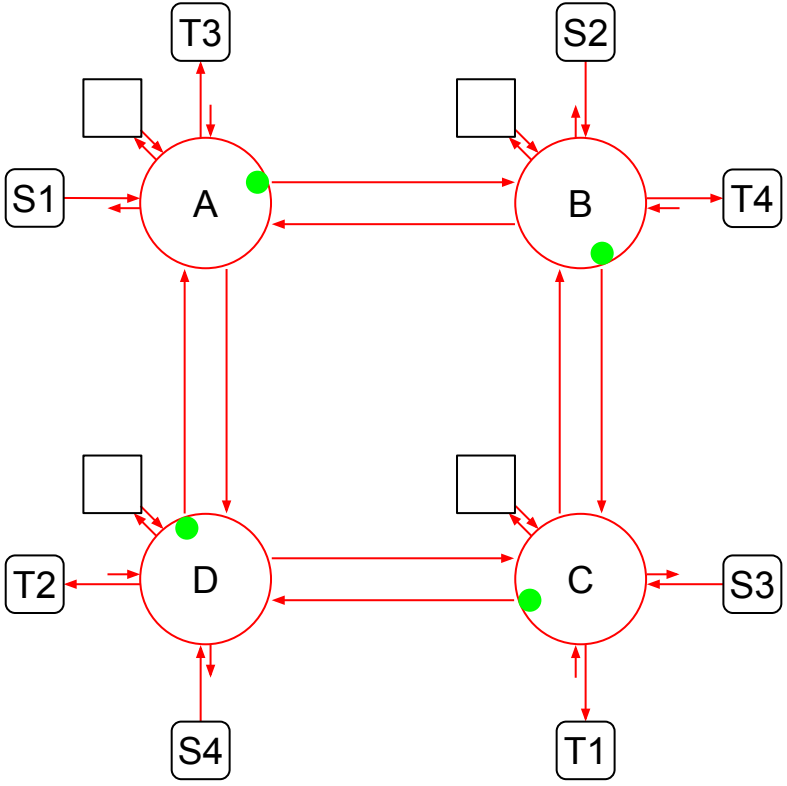
- p1 a besoin de AB puis de BC
- p2 a besoin de BC puis de CD
- p3 a besoin de CD puis de DA
- p4 a besoin de DA puis de AB

Solutions :

- limiter la taille des paquets à 1 flit
- Allouer les ressources dans l'ordre
  - les ressources partagées sont les ports de sorties
  - il faut un ordre total et une prise de décision centralisée incompatible avec le parallélisme

⇒ il faut une autre solution

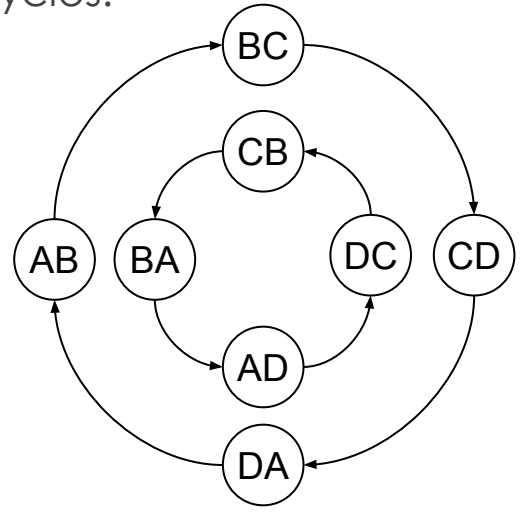
DSPIN - Analyse de dépendance



Graphe de dépendance

- noeud ressources (port de sortie)
- arc relation de dépendance

Il y a des cycles.

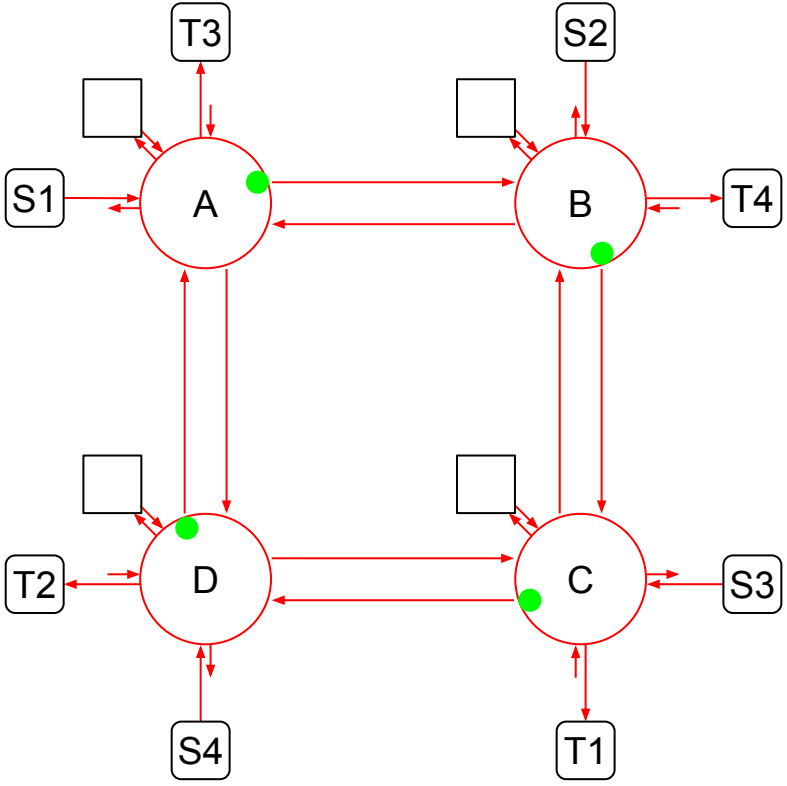


Absence de cycle ⇒ pas de deadlock
(l'inverse est faux)

C'est une condition suffisante, non nécessaire

https://link.infini.fr/m9KbV1Jf

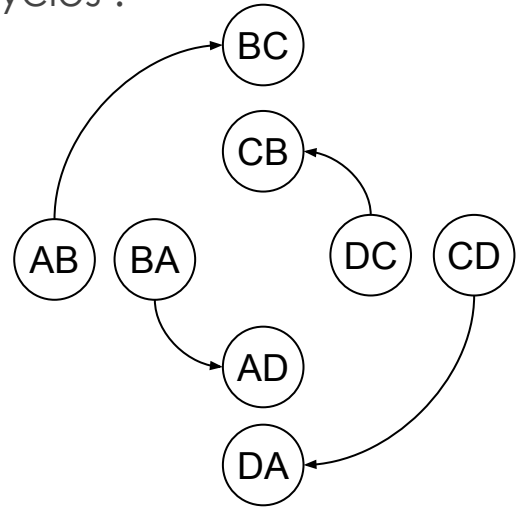
DSPIN - Analyse de dépendance



Routage X first :

- Se déplacer d'abord en X puis en Y
- ce comportement supprime des arcs

Plus de cycles !

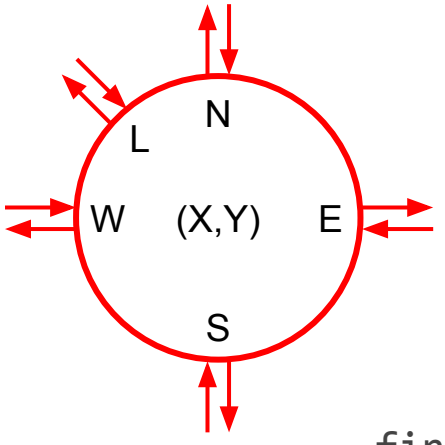


plus de deadlock

mais on suppose que les consommateurs consomme toujours

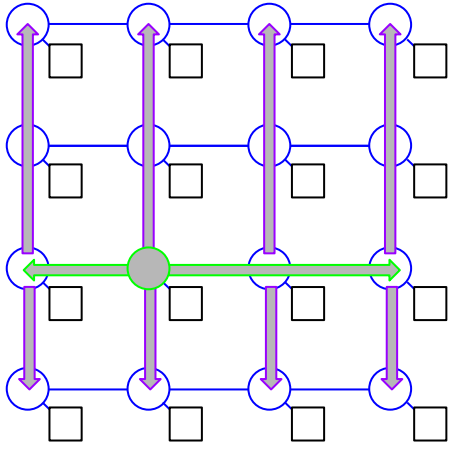
DSPIN - exemple d'une fonction de routage

une seule fonction pour tous les ports d'entrée mais simplifiée pour les frontières




```
x_first (Xd, Yd, X0, Y0)
    Si (Xd > X0) alors out = E
    sinon si (Xd < X0) alors out = W
    sinon
        si (Yd > Y0) alors out = N
        sinon si (Yd < Y0) alors out = S
        sinon out = L
        fsi
    fsi
fin x_first
```

DSPIN gère aussi un routage broadcast compatible avec le routage X-first



DSPIN - mécanisme de commutation

Paradigmes

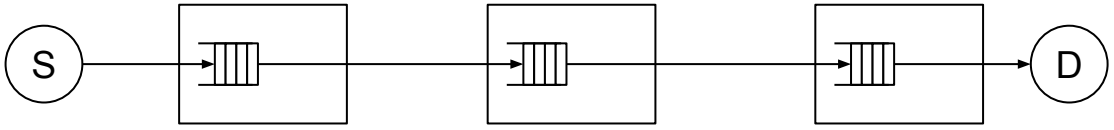
- téléphone : phase d'établissement d'un chemin et de réservation de ressource
  ⇒ après établissement : débit maximale, latence minimale
  mais débit total bas si les échanges sont courts
  ressources réservées mais non utilisée

- poste : commutation de paquet, pas de réservation de ressources
  ⇒ garantie best effort
  usage optimal du matériel mais débit et latence non garanti
  mieux adapté au échanges courts.
  C'est le modèle adopté pour les NoC

Les échanges sont très majoritairement courts, ce sont des lignes de cache,
mais quelques échanges sont très longs pour la lecture des blocs de fichiers...

DSPIN - contrôle de flux

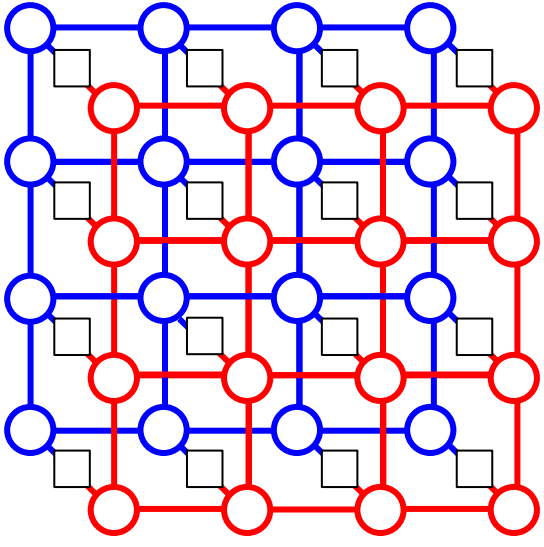
3 stratégies :



- store & forward
  - Quand un paquet entre dans un routeur, il est d'abord entièrement mémorisé(store), puis si sa destination est libre, il est transmis (forward)
    (libre = port libre et assez de place pour stocker le paquet dans le routeur suivant)
    → 😃 optimisation des ports de sortie
    → 😟 forte latence (proportionnelle à la longueur du paquet)
    → 😟 capacité importante de stockage dans les routeurs
- wormhole
  - Le routeur n'attend pas la fin du paquet entrant pour transmettre
    → allocation des ressources au plus tôt et libérer à la fin du paquet
    → 😃 faible latence + peu de stockage
    → 😟 en cas de contention sur la destination, cela se reporte sur le chemin
- cut through
  - wormhole mais avec une taille maximale pour les paquets, avec des FIFOs aussi grandes que les paquets
    → combine les avantages des 2 autres mécanismes mais encore trop de stockage

⇒ Choix pour DSPIN : worm hole mais avec des paquets bornés

Conclusion



DSPIN

- topologie : 2D-mesh
- deux meshs indépendants
- routeurs 5x5 « crossbar »
- commutation de paquets
- routage X-first
- worm hole

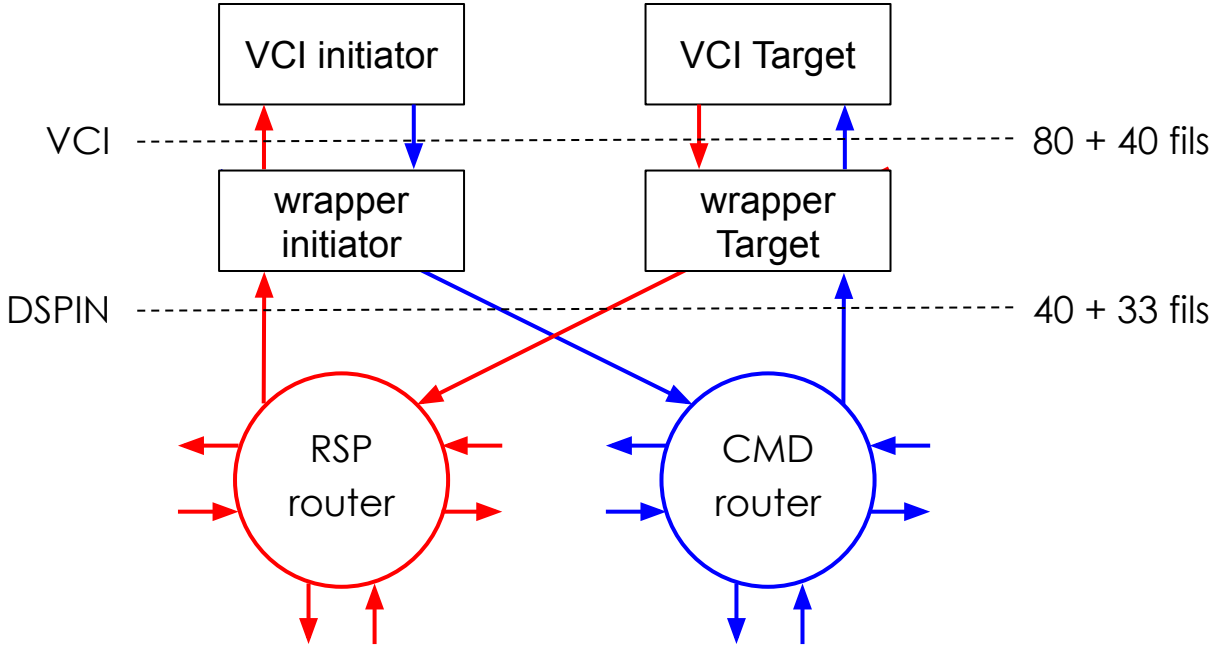
DSPIN - format des paquets

- Paquet VCi
  - multiflits
  - pas de header
  - flit très large et peu de changement
    - incrément de l'adresse
    - SRCID, PLEN, TRDID, CMD sont constants
    - WDATA, BE, EOP changent
- paquet DPSIN
  - header : 1 flit pour l'information constante
  - payload : n flit pour l'information variable

	CMD (40 bits)	RSP (33 bits)
READ	2	n+1
WRITE	n+1	2

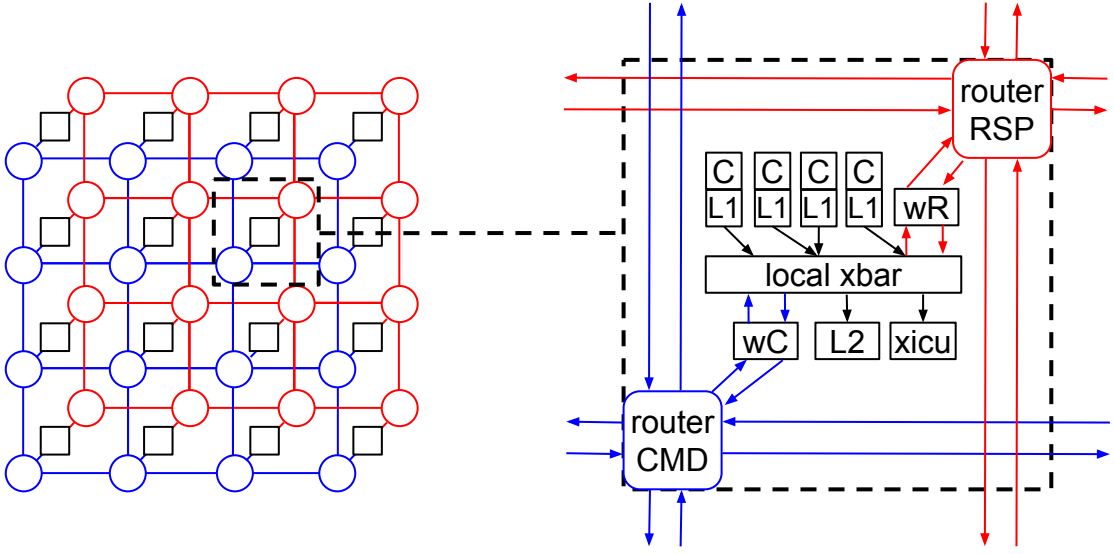
Wrapper DSPIN VCI

- Le changement de format impose l'usage de wrapper



DPSIN - positionnement -1

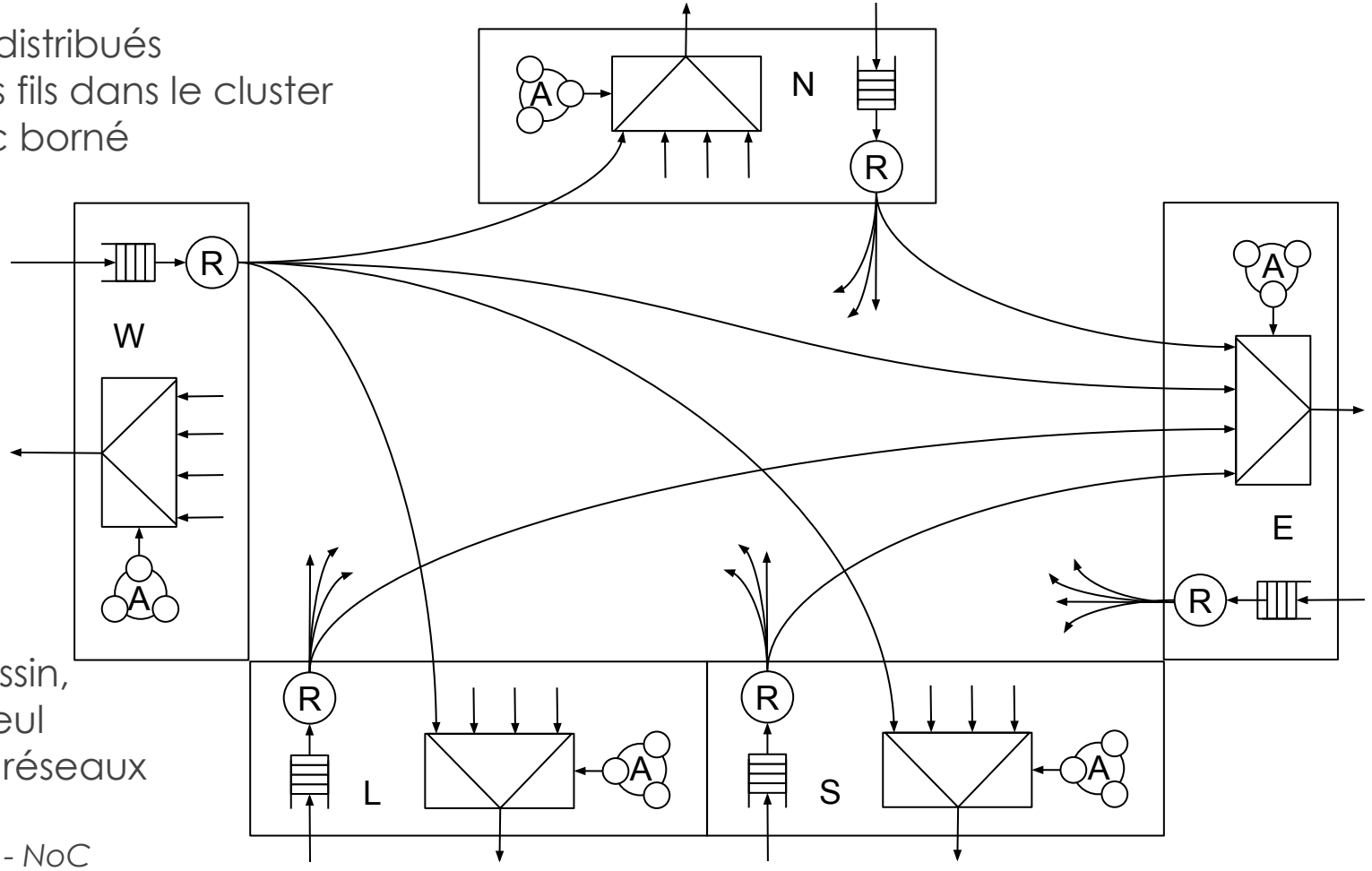
Dans les coins de chaque sous système



mais il y a des long fils et c'est plus difficile à router

DPSIN - positionnement - 2

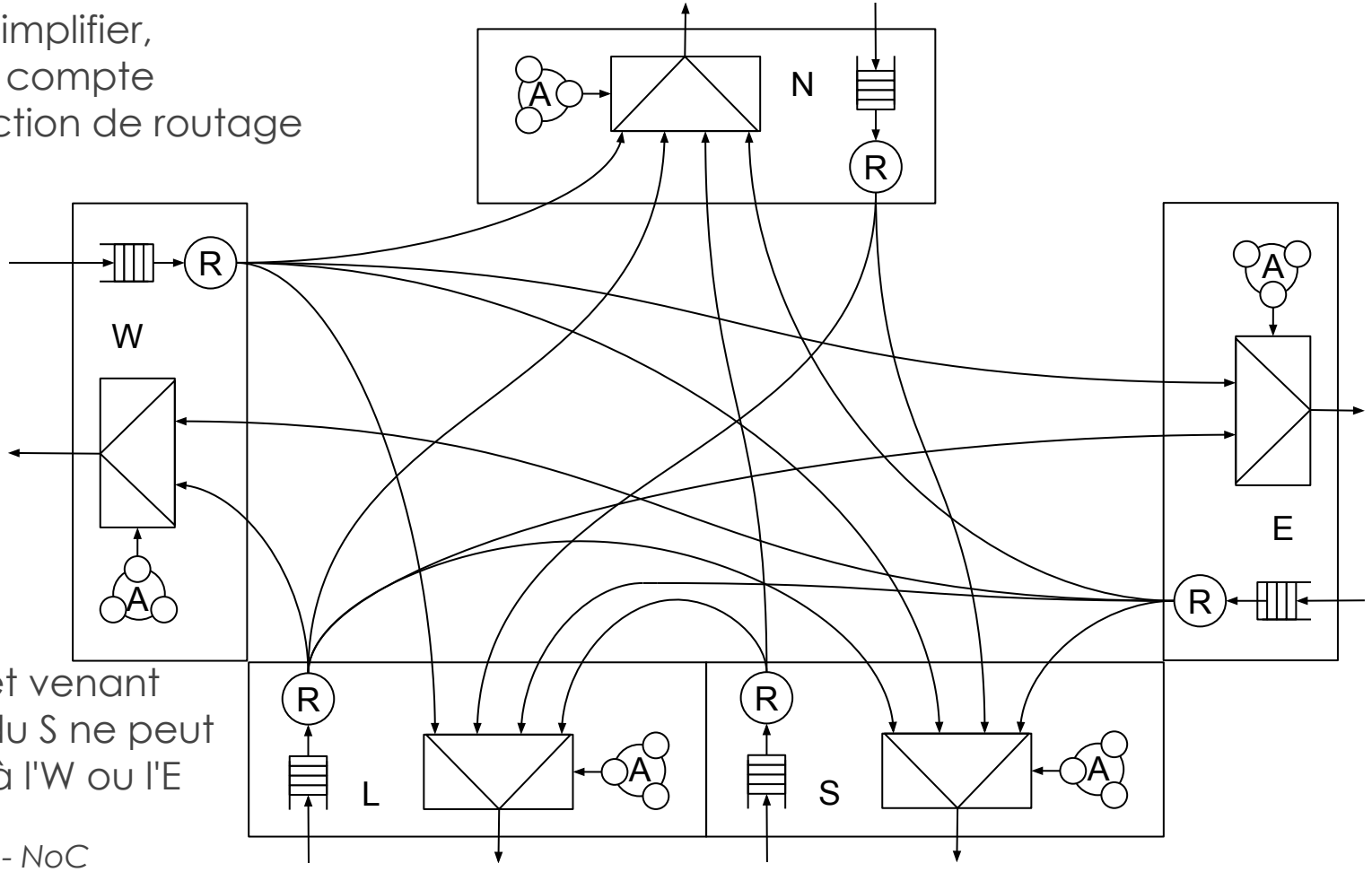
Routeurs distribués
⇒ longs fils dans le cluster
donc borné



Sur ce dessin,
il y a un seul
des deux réseaux

DPSIN - positionnement en X-first

On peut simplifier,
si on tient compte
de la fonction de routage



un paquet venant
du N ou du S ne peut
pas aller à l'W ou l'E

Conclusion

DPSIN est le NoC utilisé par TSAR

Nous verrons que 2 réseaux ne suffisent pas…

DPSIN - franchissement domaine d'horloge