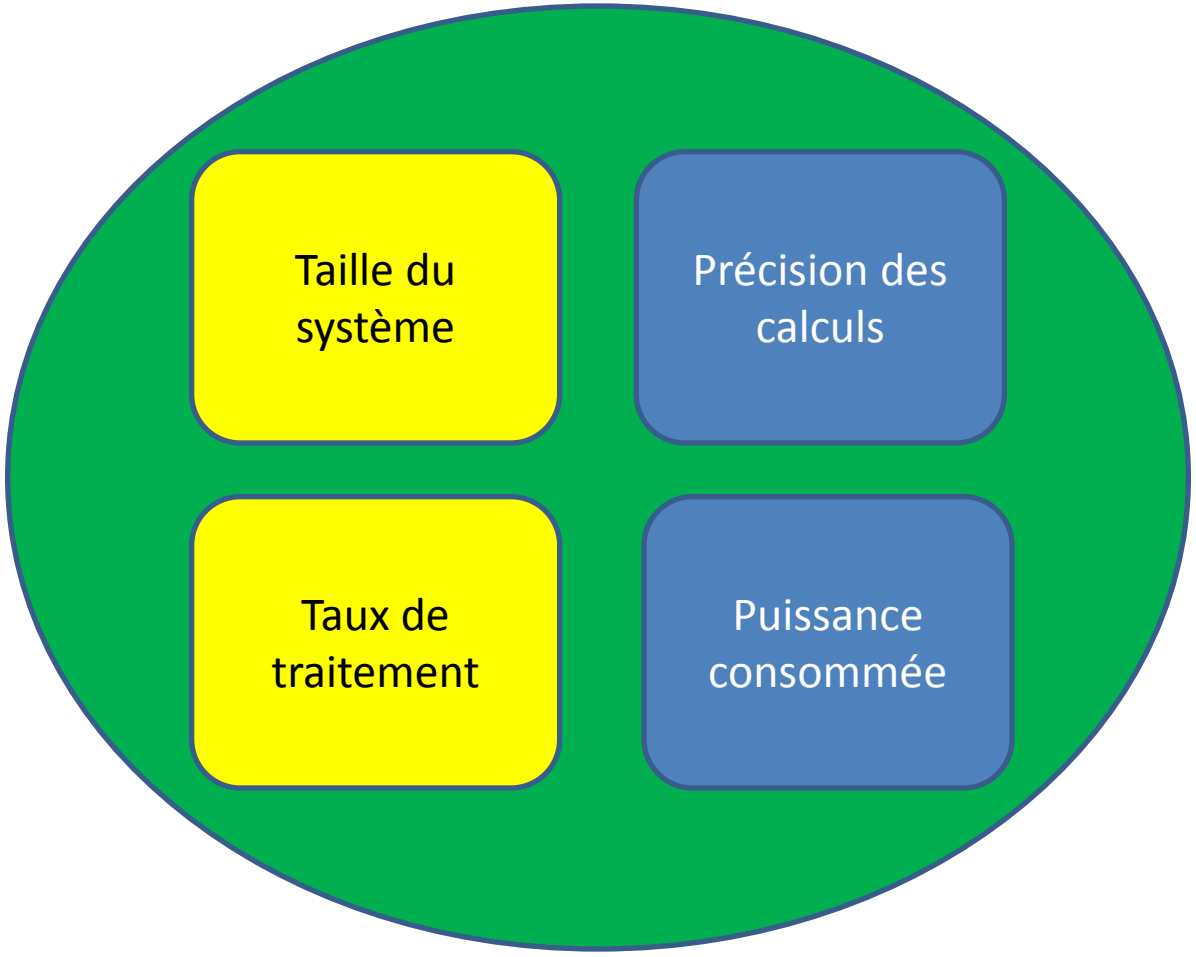
Performance de circuits numériques
Partie 2: latence, débit et surface - introduction

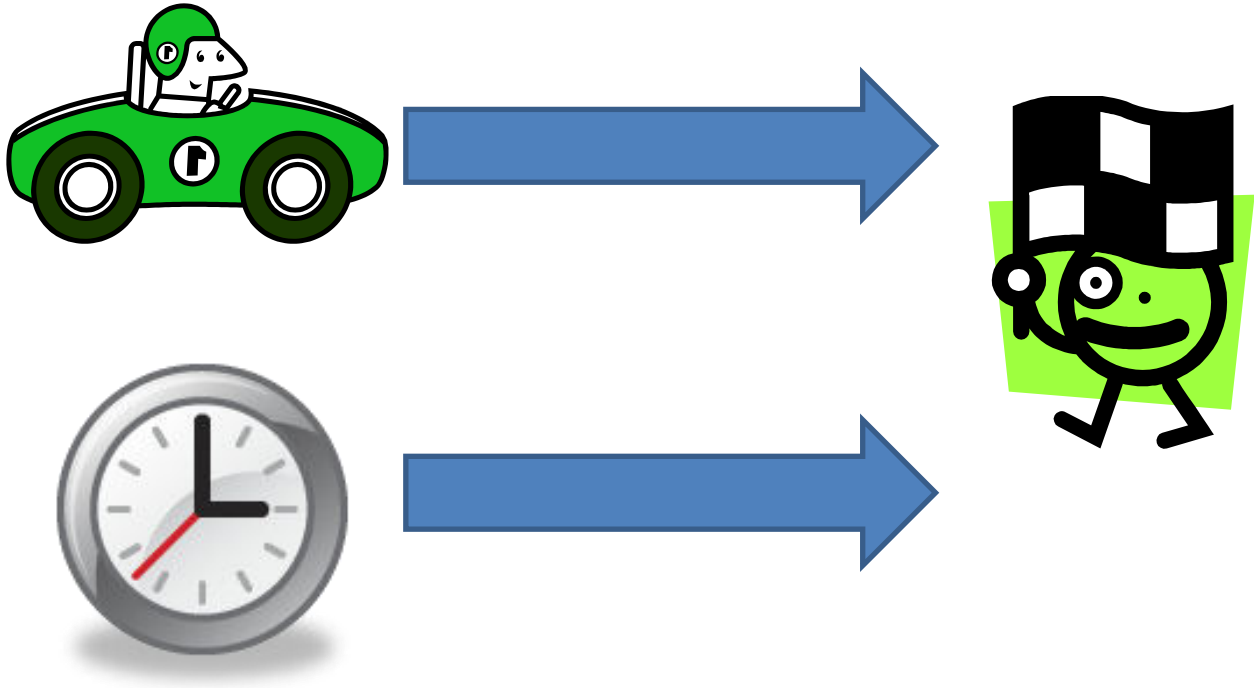
Pierre Langlois

Quatre considérations pour l'implémentation d'un système numérique



La « vitesse d'horloge », etc.

- La vitesse est le taux de déplacement d'un objet dans l'espace par unité de temps.
- Pour une horloge, il vaut mieux parler de sa *fréquence* ou de sa *période* d'horloge
- Pour un système numérique, il vaut mieux parler de sa *latence de calcul* et de son *débit d'information* (ou taux de traitement).
- Par « haute performance » d'un système numérique, on veut souvent dire *haut débit*, et parfois *faible latence*.



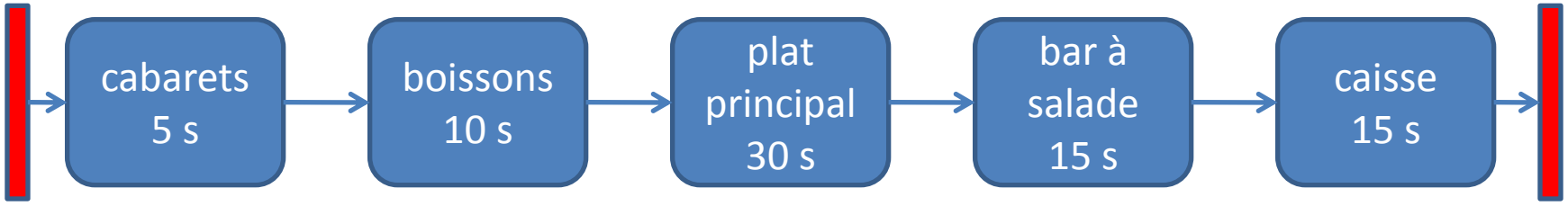
Impact de l'architecture sur la latence et le débit

- Considérons une cafétéria avec 5 stations.
- Supposons:
  - il y a un seul client dans la ligne à la fois
  - chaque client passe par chaque station
  - un client ne peut pas prendre son cabaret tant que le client précédent n'a pas fini à la caisse

- Période = 75 s; fréquence = 13.3 mHz
- Latence:

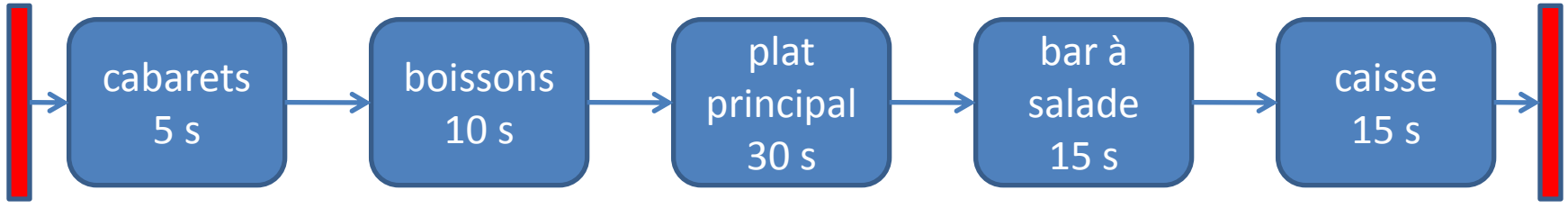
  1 cycle = 75 secondes pour servir un client
- Débit:

  1 client / 75 s × 3600 s/h = 48 clients par heure



Latence et débit

- La *latence de calcul* est le temps nécessaire pour produire un résultat.
  - C'est le temps entre le moment où une donnée est disponible et le moment où le résultat qui dépend de cette donnée est disponible à son tour.
  - Courte latence = meilleure performance.
  - La latence est souvent exprimée en nombre de cycles d'horloges.
  - Une fréquence d'horloge plus élevée correspond à une latence plus courte.
  - Le parallélisme n'a pas d'influence sur la latence.
- Le *débit d'information* est le nombre de résultats produits par unité de temps.
  - On suppose qu'une quantité suffisante de données est disponible à l'entrée du système pour le garder toujours actif.
  - Grand débit = meilleure performance.
  - Le débit est exprimé en nombre de résultats par seconde.
  - Une fréquence d'horloge plus élevée correspond à un débit plus grand.
  - Le parallélisme augmente le débit.



Impact de l'architecture sur la latence et le débit

- Considérons une cafétéria avec 5 stations.
- Supposons:
  - Il peut y avoir un client à chaque station, mais pas plus d'un client par station.

- Période = 30 s; fréquence = 33.3 mHz
- Latence

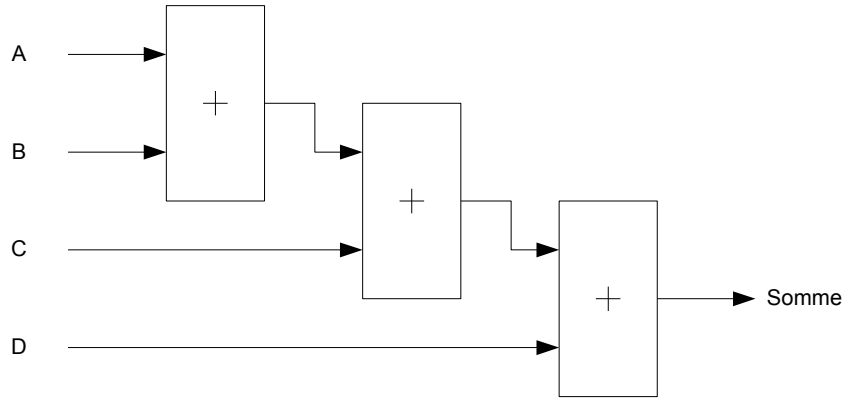
  5 cycles = 150 secondes pour servir un client
- Débit

  1 client / 30 s × 3600 s/h = 120 clients par heure

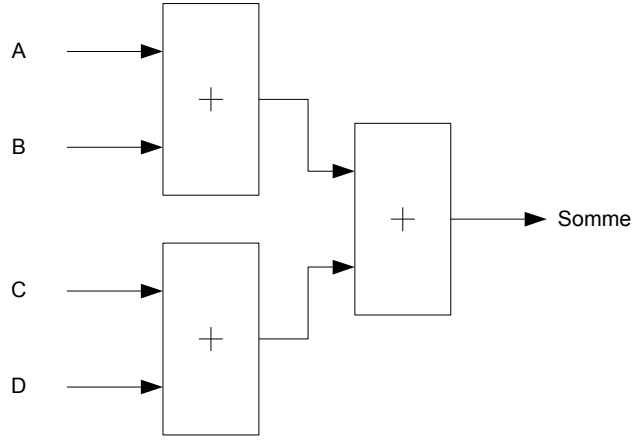
Impact de l'architecture sur la latence

- Certains processeurs ne traitent pas un flux important de données, mais traitent plutôt des données ponctuelles à intervalles irréguliers.
- Dans un tel cas, il est plus important de réduire la latence que d'augmenter le débit. Exemples:
  - communications bidirectionnelles;
  - transactions financières en bourse;
  - recherche web.
- Stratégies:
  - Réduire le délai sur le chemin critique.
  - Réorganiser l'ordre dans lequel les calculs sont faits.
  - Paralléliser les calculs sur plusieurs unités.
  - Éviter le pipeline.

```
Somme <= A + B + C + D;
```



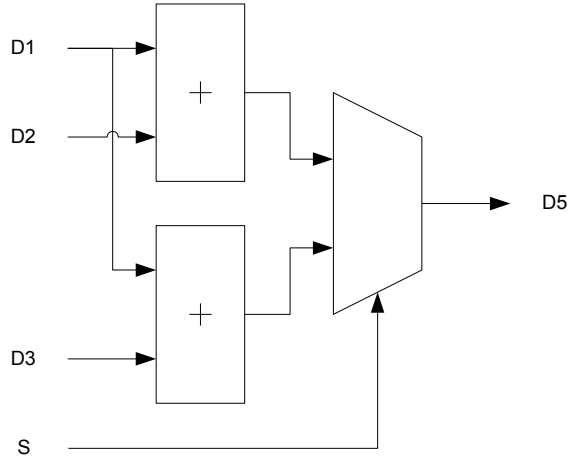

```
Somme <= (A + B) + (C + D);
```



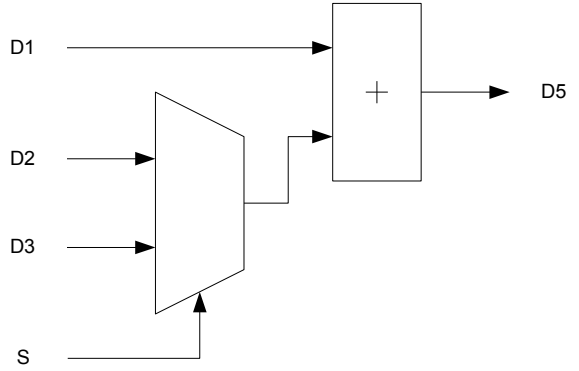
Impact de l'architecture sur la surface
Partage de ressources

- L'ordre des opérations peut minimiser la surface utilisée. Voir l'exemple:
  - Le chemin critique est le même pour les deux exemples, mais la version de droite n'utilise qu'un seul additionneur.
- Utiliser des stratégies contraires à celles utilisées pour maximiser le débit: éliminer le parallélisme.
- Au lieu d'instancier plusieurs composantes identiques pour effectuer des calculs, on en n'utilise qu'une seule et on en contrôle l'accès à l'aide d'une unité de contrôle qui implémente une machine à états.

```
process (D1, D2, D3, S)
begin
   if S = '1' then
      D5 <= D1 + D2;
   else
      D5 <= D1 + D3;
   end if;
end process;
```




```
process (D1, D2, D3, S)
variable t : type-des-Di;
begin
   if S = '1' then
      t := D3;
   else
      t := D2;
   end if;
   D5 <= D1 + t;
end process;
```

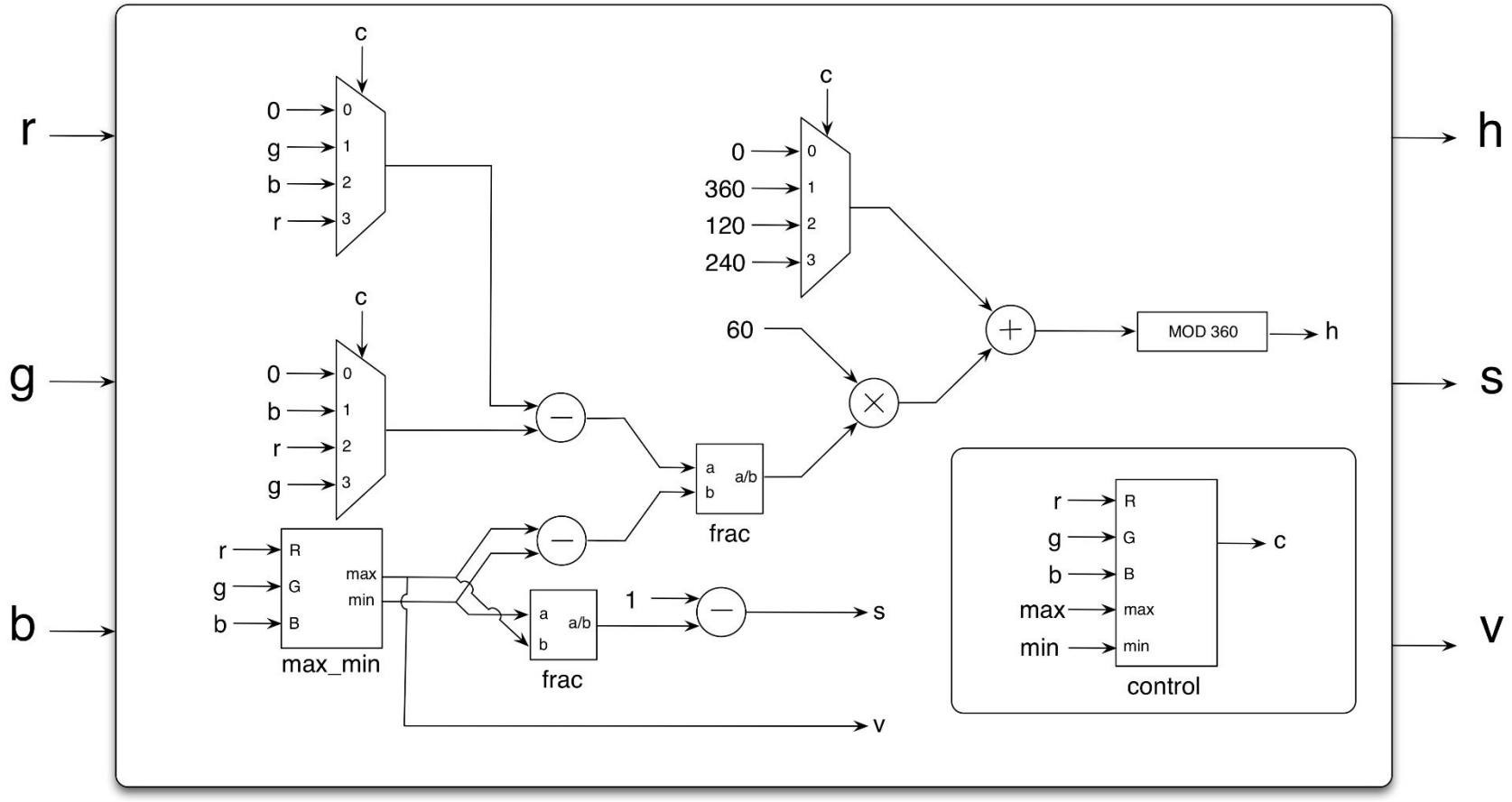


Impact de l'architecture sur la surface
Partage de ressources

$$h = \begin{cases} 0 & \text{Si} \quad \max = \min \\ 360 + 60 \times \frac{g-b}{\max-\min} \bmod 360 & \text{Si} \quad \max = r \\ 120 + 60 \times \frac{b-r}{\max-\min} & \text{Si} \quad \max = g \\ 240 + 60 \times \frac{r-g}{\max-\min} & \text{Si} \quad \max = b \end{cases}$$

$$s = \begin{cases} 0 & \text{Si } \max = 0 \\ \frac{\max-\min}{\max} & \text{Autrement} \end{cases}$$

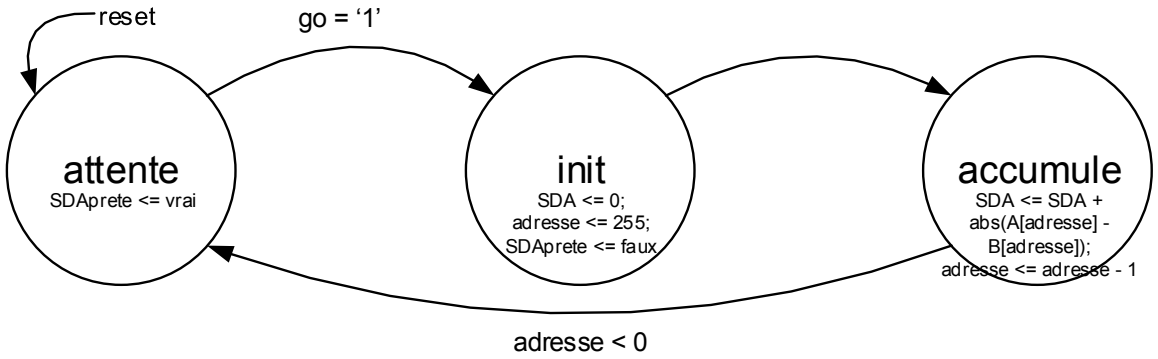
$$v = \max$$

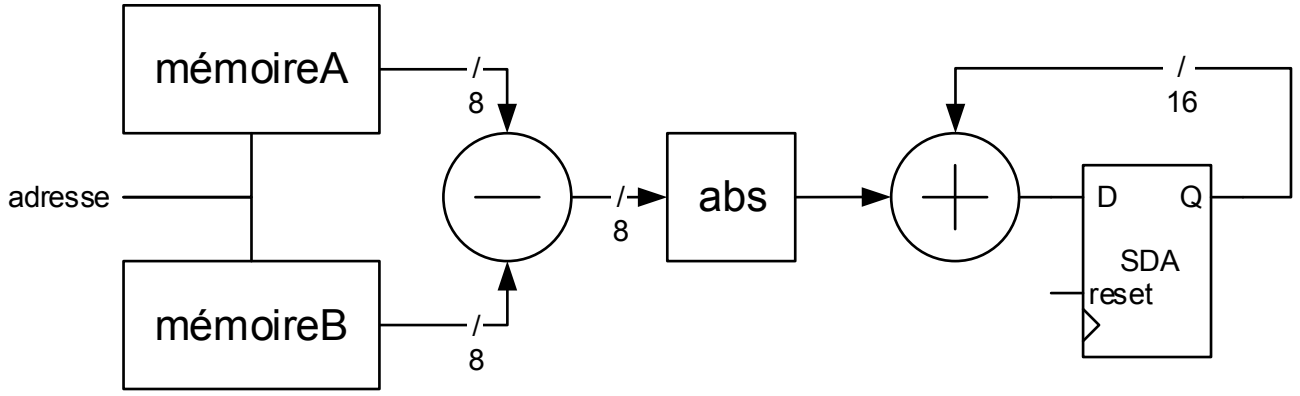


Impact de l'architecture sur la surface
Étaler les calculs dans le temps

- Les FPGA offrent une grande quantité de structures de calculs. On les exploite en général en utilisant le parallélisme au maximum pour maximiser le débit.
- Parfois par contre on doit minimiser la surface utilisée. On doit alors réutiliser une même structure de calcul pour plusieurs opérations.
- Exemple: calcul de la somme des différences absolues entre les éléments de deux vecteurs.

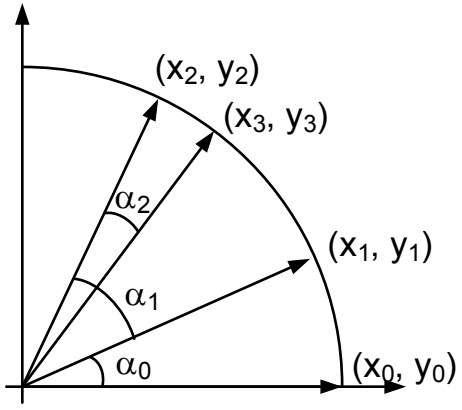
$$SDA = \sum_{k=0}^{255} \left| A_k - B_k \right|$$

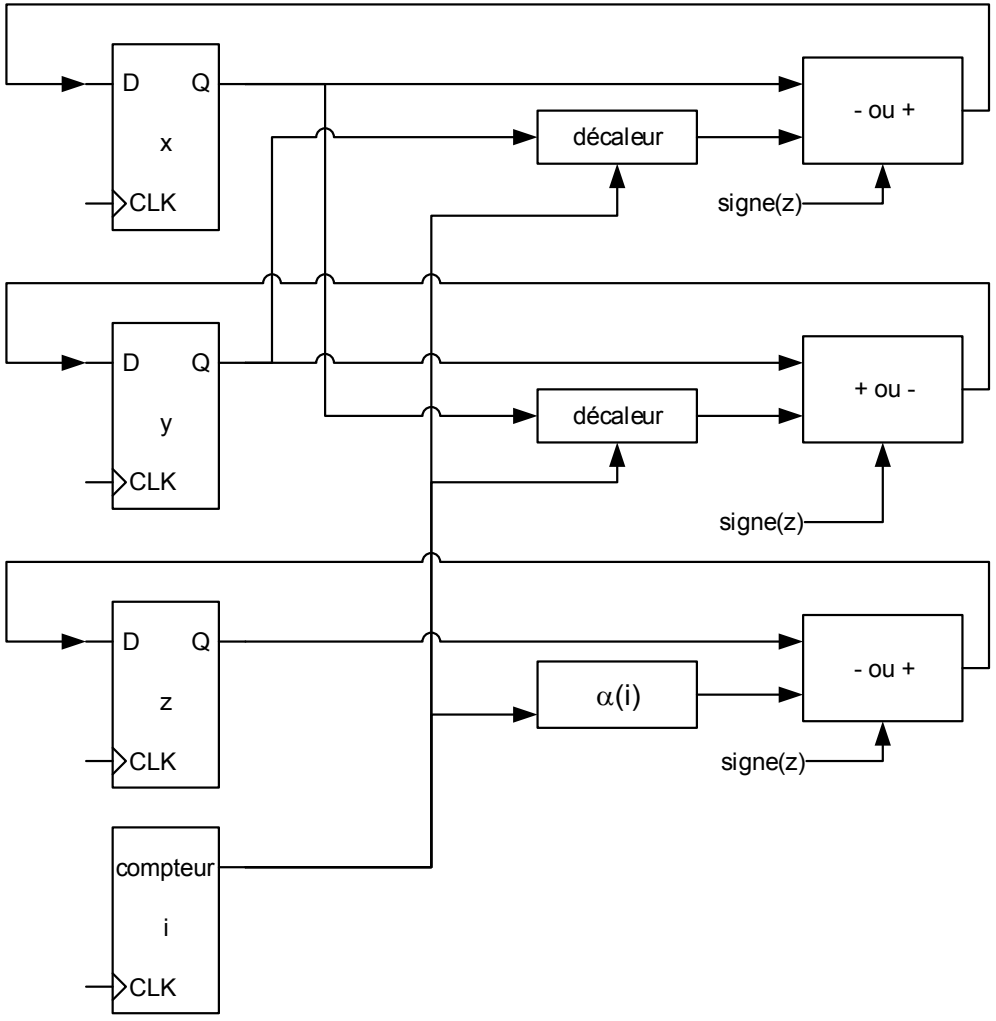




Impact de l'architecture sur la surface
Étaler les calculs dans le temps

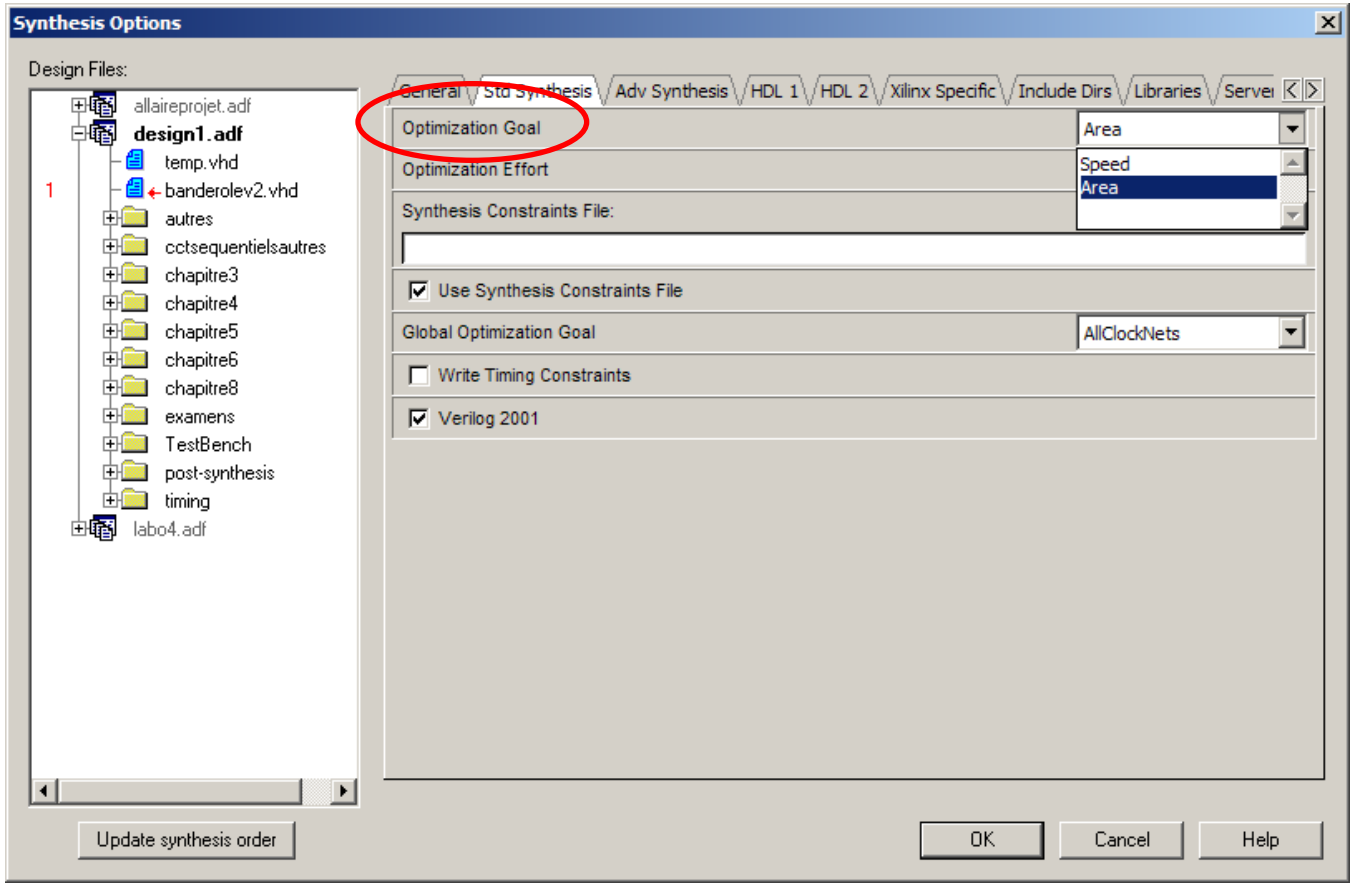
- Les FPGA offrent une grande quantité de structures de calculs. On les exploite en général en utilisant le parallélisme au maximum pour maximiser le débit.
- Parfois par contre on doit minimiser la surface utilisée. On doit alors réutiliser une même structure de calcul pour plusieurs opérations.
- Exemple: CORDIC





Favoriser la surface ou le débit lors de la synthèse

- Un synthétiseur peut parfois faire le meilleur choix indépendamment de la description du design.



En conclusion …

- En règle générale, plus une décision de design est prise à un haut niveau d'abstraction et plus son impact sera grand sur la latence, le débit et la surface d'un système.
- Une bonne décision prise à un bas niveau d'abstraction ne peut pas en général compenser pour une mauvaise décision prise à un haut niveau d'abstraction.