Conception de Systèmes Numériques sur FPGA

Décomposition des Systèmes Numériques

Yann Thoma

Reconfigurable and Embedded Digital Systems Institute
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

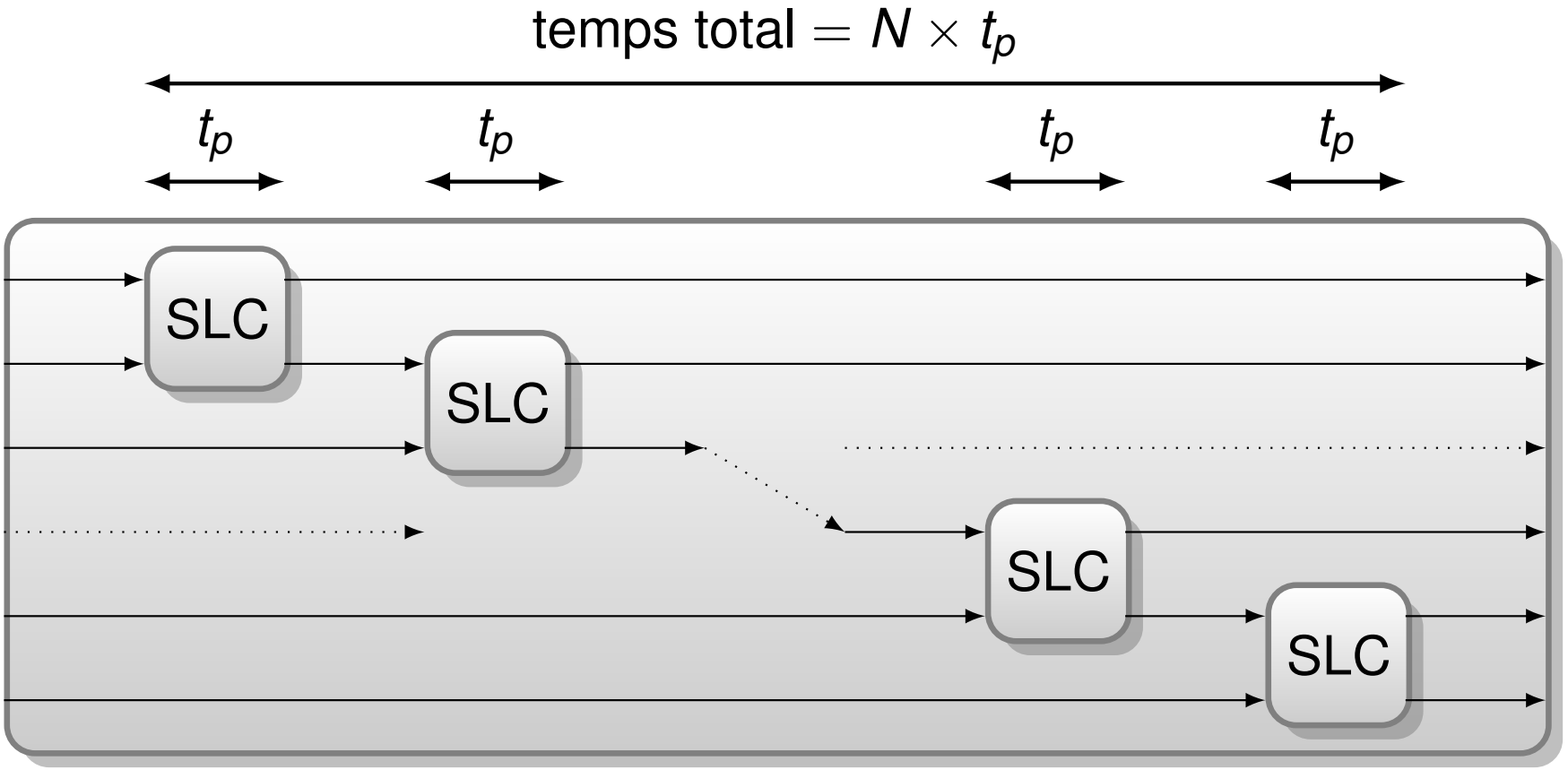
Février 2017

Décomposition des circuits logiques

- Réalisation de systèmes de traitement de données:
  - Il s'agit souvent d'opérations combinatoires complexes
- Deux structures possibles:
  - Solution purement combinatoire : décomposition spatiale
  - Solution séquentielle : décomposition temporelle

Décomposition spatiale

- La complexité est décomposée en modules de base combinatoires qui sont ensuite interconnectés dans l'espace

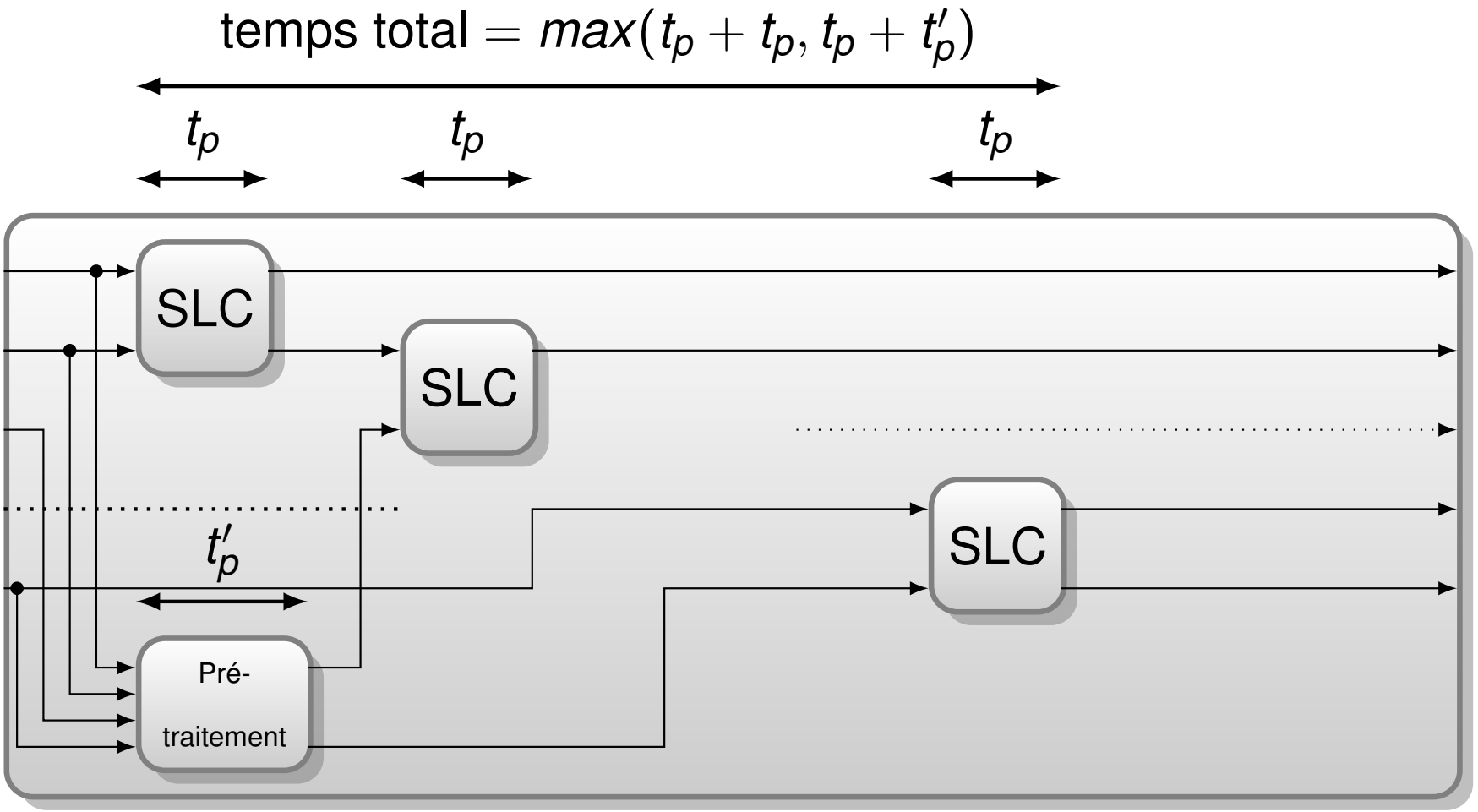


Décomposition spatiale: optimisation

- L'optimisation d'une solution spatiale nécessite de connaître les chemins critiques (le plus long)
- Dans le cas de l'addition, le chemin critique est le report. Le résultat de l'addition est valide seulement lorsque le dernier report est calculé
- La solution est de trouver un ou plusieurs chemins parallèles qui permettent de raccourcir le temps de calcul. Cela nécessite du matériel supplémentaire.
  - Pour l'addition il s'agit de la technique d'anticipation du report (carry look-ahead)

Décomposition spatiale: optimisation

- Un ou plusieurs modules en début de traitement permettent de supprimer le chaînage des modules

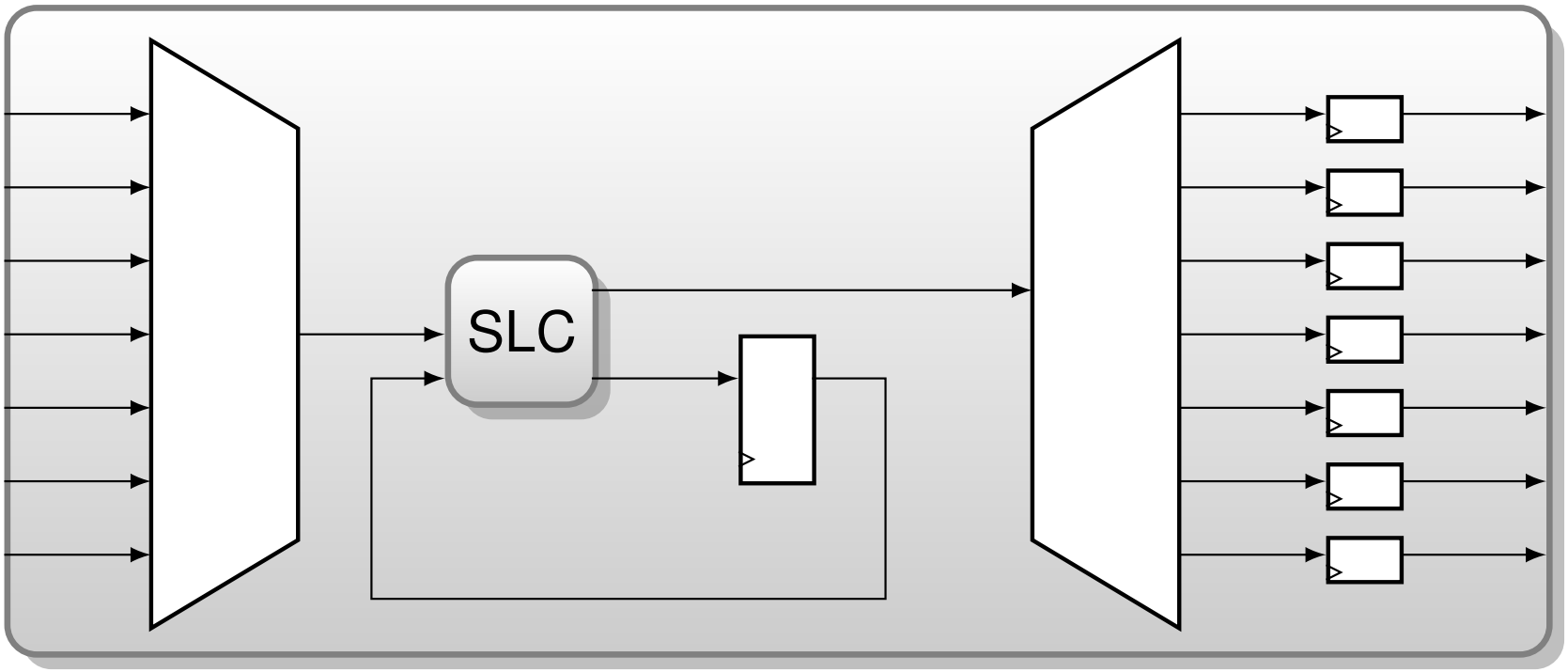


Décomposition spatiale

- Les logiciels de synthèse sont capable de certaines optimisations
  - Mais pas toutes
- Un bon concepteur de systèmes numériques doit pouvoir modifier son design pour respecter certaines contraintes
  - Fréquence de fonctionnement
  - Débit/latence

Décomposition temporelle

- Le système est composé d'un seul module de base qui est utilisé séquentiellement pour réaliser toutes les étapes nécessaires au calcul



Décomposition temporelle: optimisation

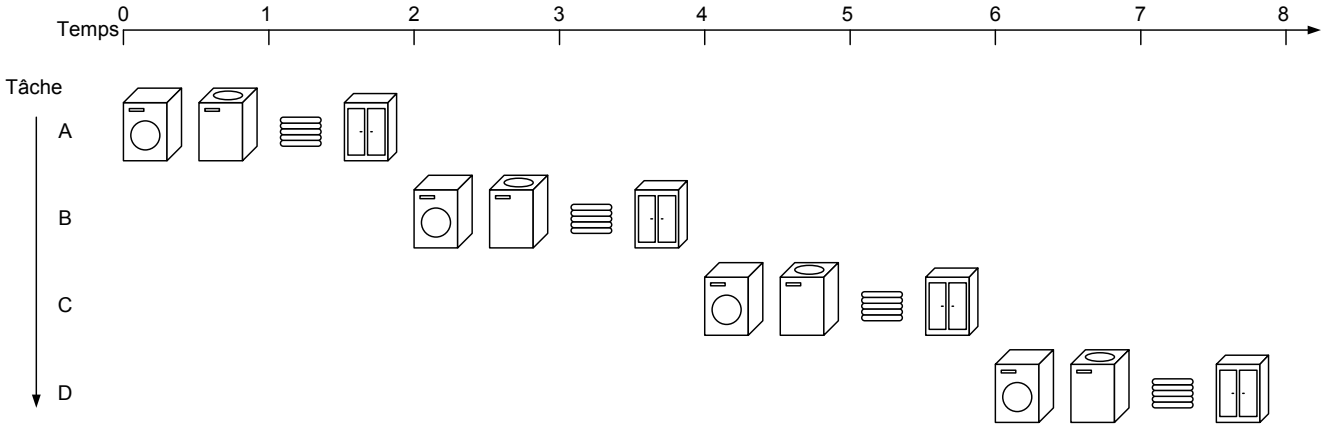
- La partie principale à optimiser : module combinatoire
  - Ce module est implanté une seule fois, il est possible de paralléliser le traitement (augmentation de la quantité de matériel)
  - Il est aussi possible d'utiliser un bloc RAM du PLD
    - Solution parfois plus rapide que de la logique
  - Exemple : module de base additionneur 4 bits
    - réalisation parallèle de l'additionneur (ne pas utiliser une structure de chaînage d'additionneurs 1 bit)
- Concernant les registres, seul un changement de technologie permet d'améliorer les performances

Pipelining en machine: Exemple du lavage

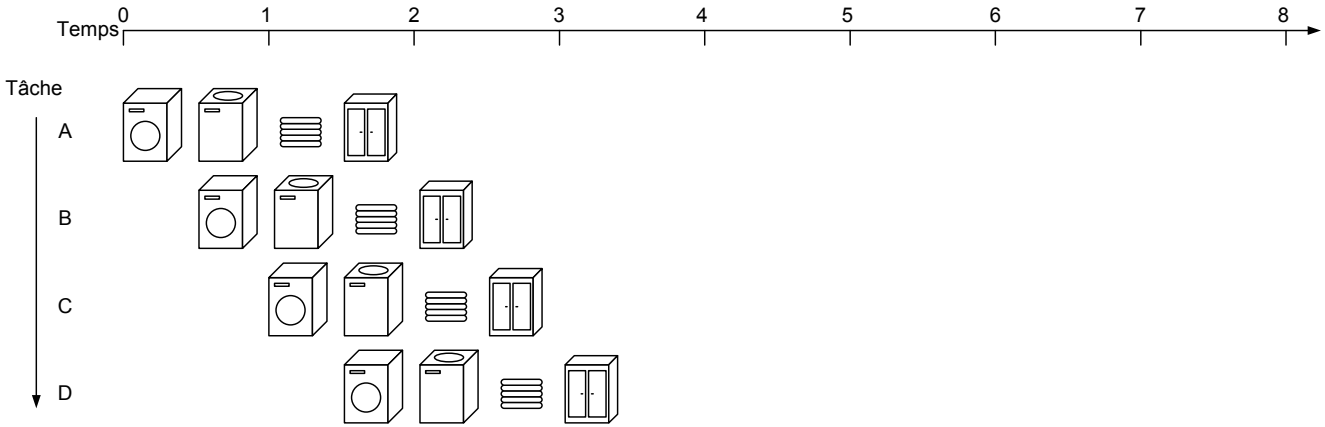
- Georges, Roger, Albert et Gontran doivent laver leurs affaires
- Le lavage prend 30 minutes
- Le séchage prend 30 minutes
- Le pliage prend 30 minutes
- Le rangement prend 30 minutes

Exemple: lavage

- Version simple



- Version pipeline

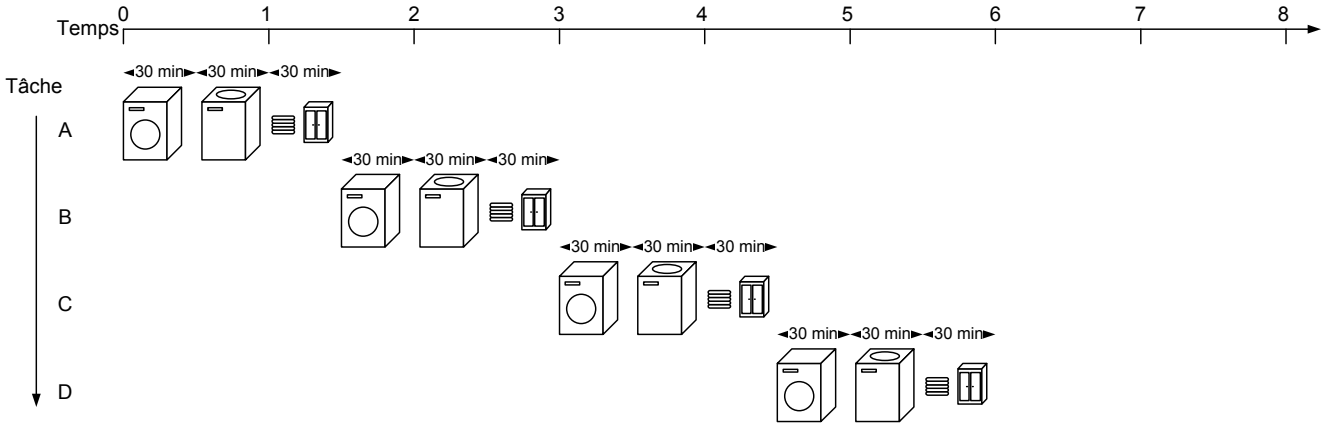


Exemple: lavage

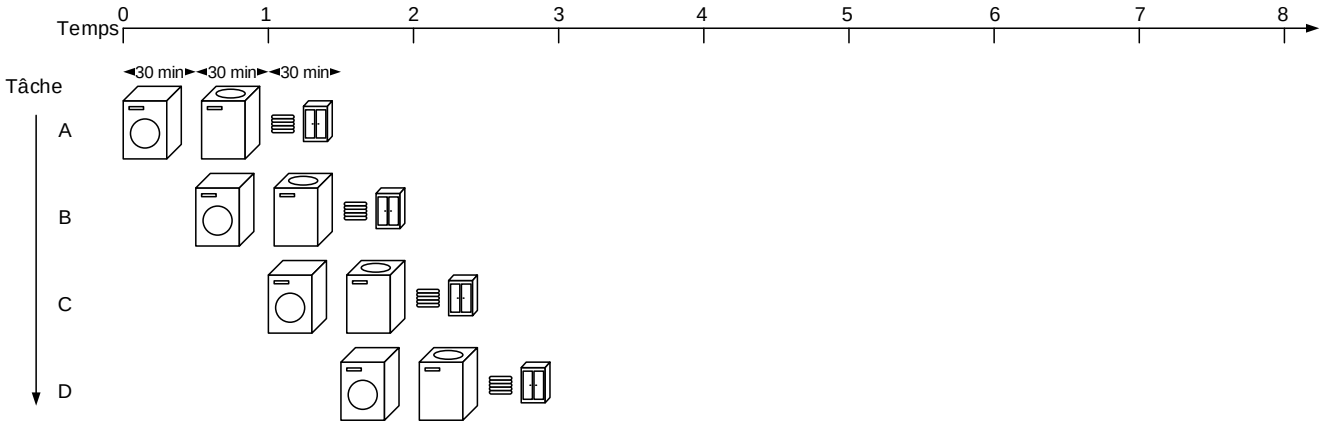
- Soyons réalistes!!!!
- Georges, Roger, Albert et Gontran doivent laver leurs affaires
- Le lavage prend 30 minutes
- Le séchage prend 30 minutes
- Le pliage prend 15 minutes
- Le rangement prend 15 minutes

Exemple: lavage

- Version simple

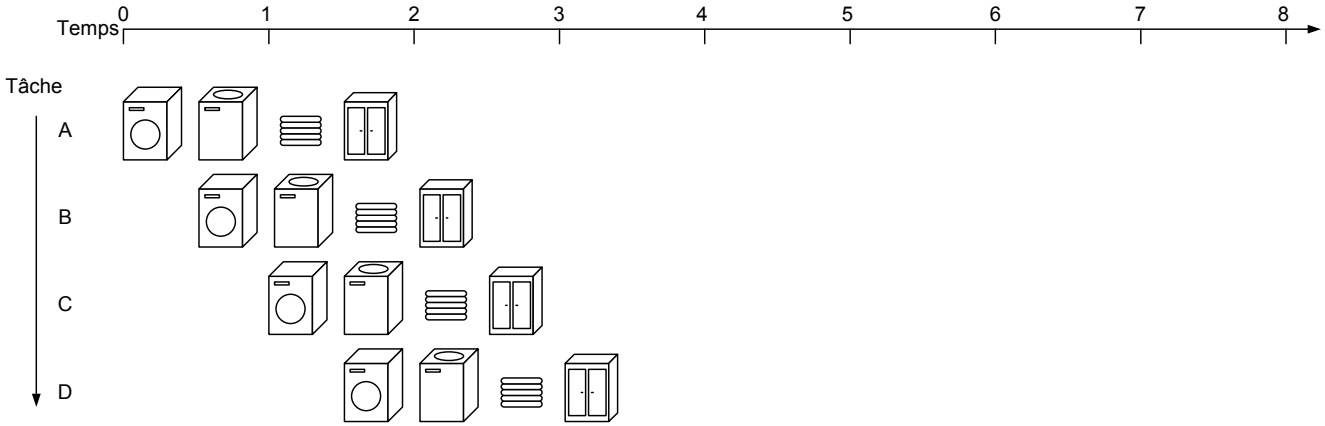


- Version pipeline

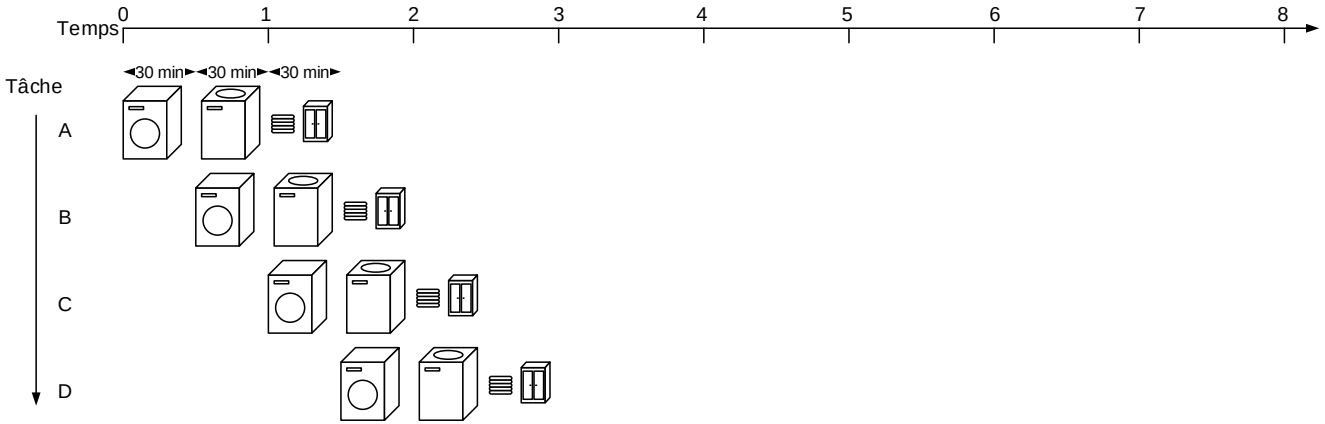


Exemple: lavage (comparaison)

- Version pipeline 1



- Version pipeline 2



Analyse

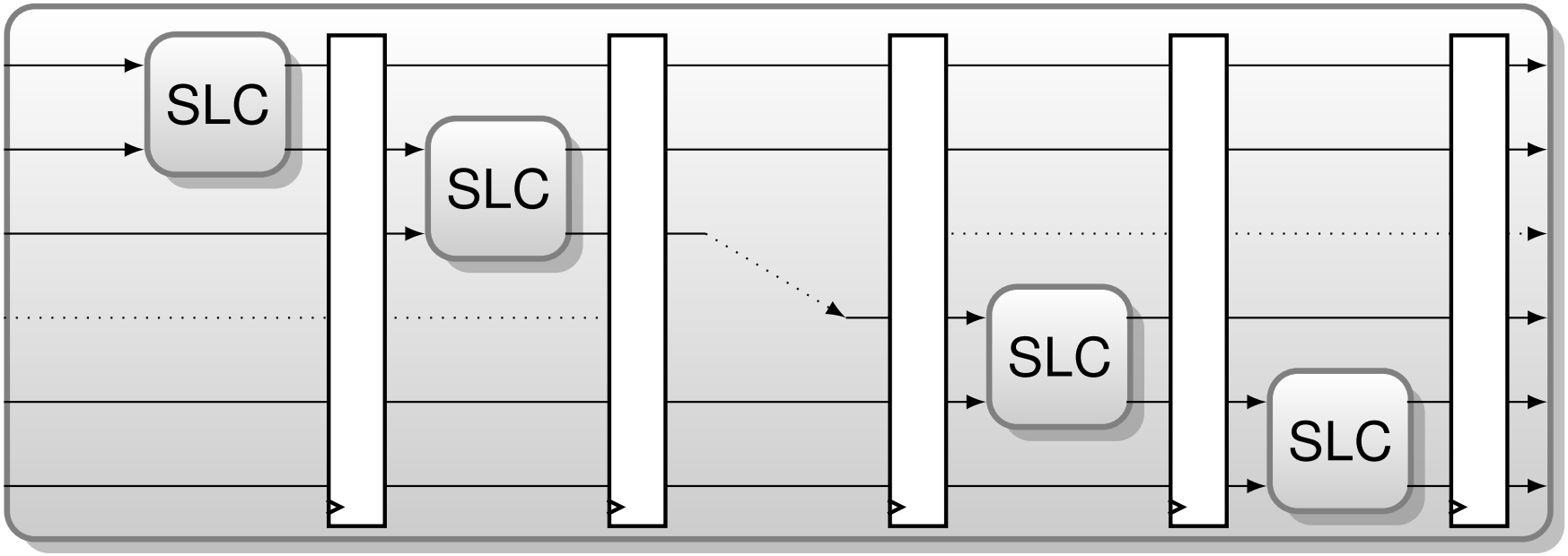
- Le pipeline n'améliore pas la latence, mais le débit
  - Un seul traitement n'est pas plus rapide
  - Mais le débit sur plusieurs traitements oui
  - Dans l'exemple:
    - Temps d'un traitement complet: 1h30
    - Débit: Une machine par 30 minutes
- Plusieurs tâches exploitant différentes ressources exécutées en parallèle
- Accélération potentielle = nombre d'étages du pipeline
- La fréquence est limitée par l'étage le plus lent
- Des étages non équilibrés réduisent la vitesse

Structure pipelinée

- L'objectif est d'augmenter le débit de traitement. Le système combinatoire est découpé en petites tranches afin de pouvoir utiliser une fréquence plus élevée.
- Réalisation :
  - Fractionner le système combinatoire (décomposition spatiale) en tranches de calculs. Chaque tranche nécessite un temps beaucoup plus court (t_p)
  - Mémoriser les résultats intermédiaires entre chaque tranche du calcul
  - A chaque période, une étape est calculée. Il faudra N périodes d'horloge pour réaliser une opération (calcul)
  - Lorsque le pipeline est amorcé, il y a un résultat fourni à chaque période d'horloge

Structure pipelinée

- Schéma pour une réalisation avec une structure "pipeline"
  - Ajout des registres intermédiaires



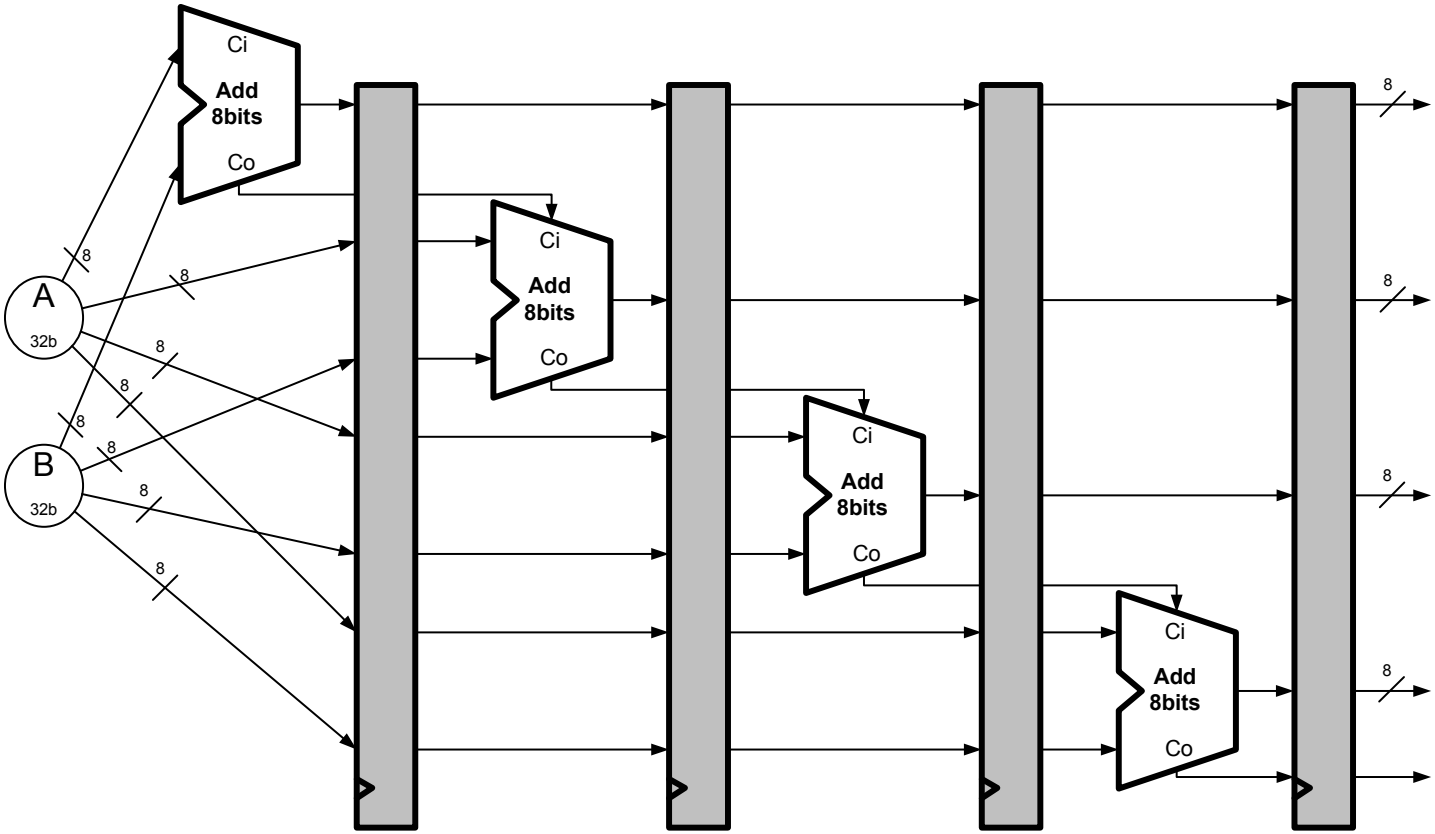
Structure pipelinée

- Cette solution est très performante dans le cas de :
  - Traitement utilisé de nombreuses fois
  - Nombre important de données à traiter
- Permet d'atteindre des débits de traitement très élevés
- Implique un temps de latence important
  - Lié au nombre d'étages du pipeline
- Cette structure utilise passablement de matériel
  - modules combinatoires + registres intermédiaires

Exemple: Structure pipelinée pour l'addition

- Additionner deux nombres de 32 bits non signés
- On dispose d'additionneurs 8 bits avec $tp_{ADD} = 10ns$ et de registres 8 bits avec $tp_{REG} = 3ns$ (8 flip-flops)
- On souhaite disposer du plus grand débit de traitement possible.
- Solutions possibles :
  - Solution combinatoire: $N * tp_{ADD} = 4 * 10ns = 40ns$
  - Solution séquentielle : pas performante
  - Solution avec pipeline : $tp_{ADD} + tp_{REG} = 13ns$
    - $\Rightarrow$ par contre il faudra 4 périodes pour amorcer le pipeline

Structure pipelinée pour l'addition

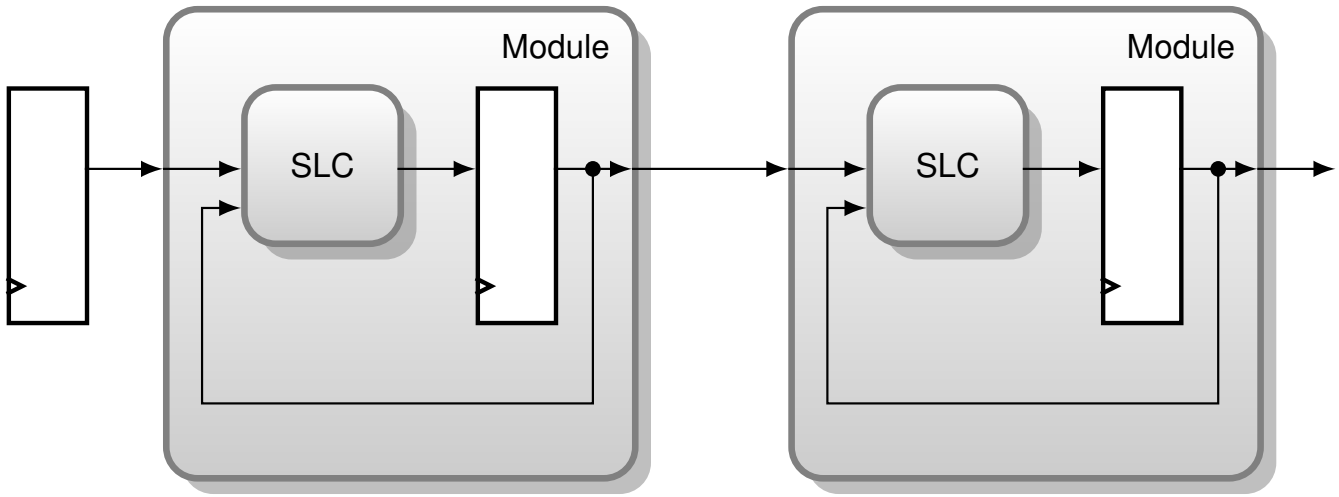


Amélioration des performances

- Egaliser au maximum les chemins combinatoires
- Si la fréquence de fonctionnement est trop faible
  - Il y a un ou plusieurs chemins combinatoires trop long
  - Rechercher les chemins critiques entre deux flip-flops
    - $\Rightarrow$ utiliser l'analyse de timing statique des outils PR
- Solution :
  - Déplacer le bloc combinatoire
  - Anticiper l'opération combinatoire
  - Couper le bloc combinatoire en plusieurs parties selon le principe d'une structure pipeline

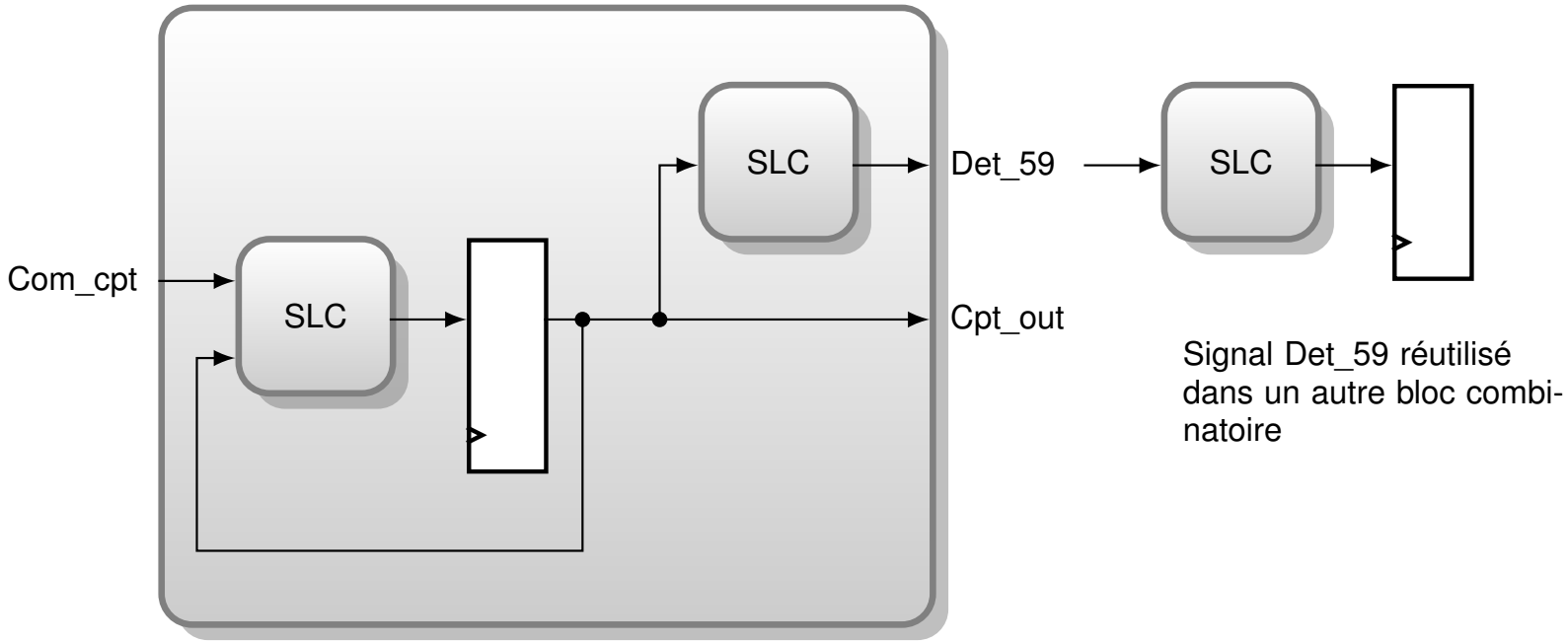
Maîtriser les performances

- Bonne pratique: machines de Moore (voir Medvedev)
  - Sorties d'un module VHDL correspond toujours à la sortie d'une bascule
  - Optimisation des timings possible séparément pour chaque module



Exemple: report d'un compteur

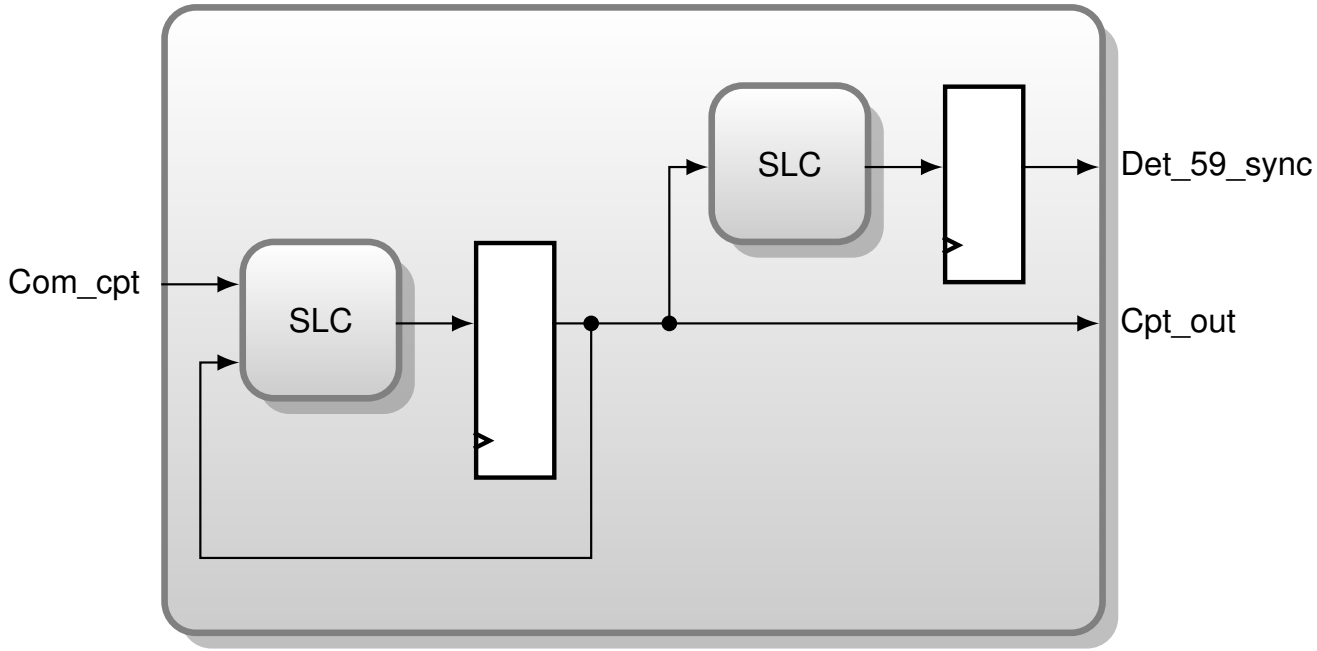
- Décomposition de la structure du compteur



- Analyse:
  - La sortie Det_59 a un délai de propagation non négligeable après l'activation de l'horloge (flanc).
  - Les blocs combinatoires utilisant ce signal seront retardés d'autant

Exemple: report d'un compteur

- Solution: post-synchroniser le signal Det_59



- Analyse:
  - La sortie Det_59_sync a un délai de propagation très faible par rapport au flanc de l'horloge.
  - Mais le signal est retardé d'une période d'horloge !

Exemple: report d'un compteur

Description VHDL du report

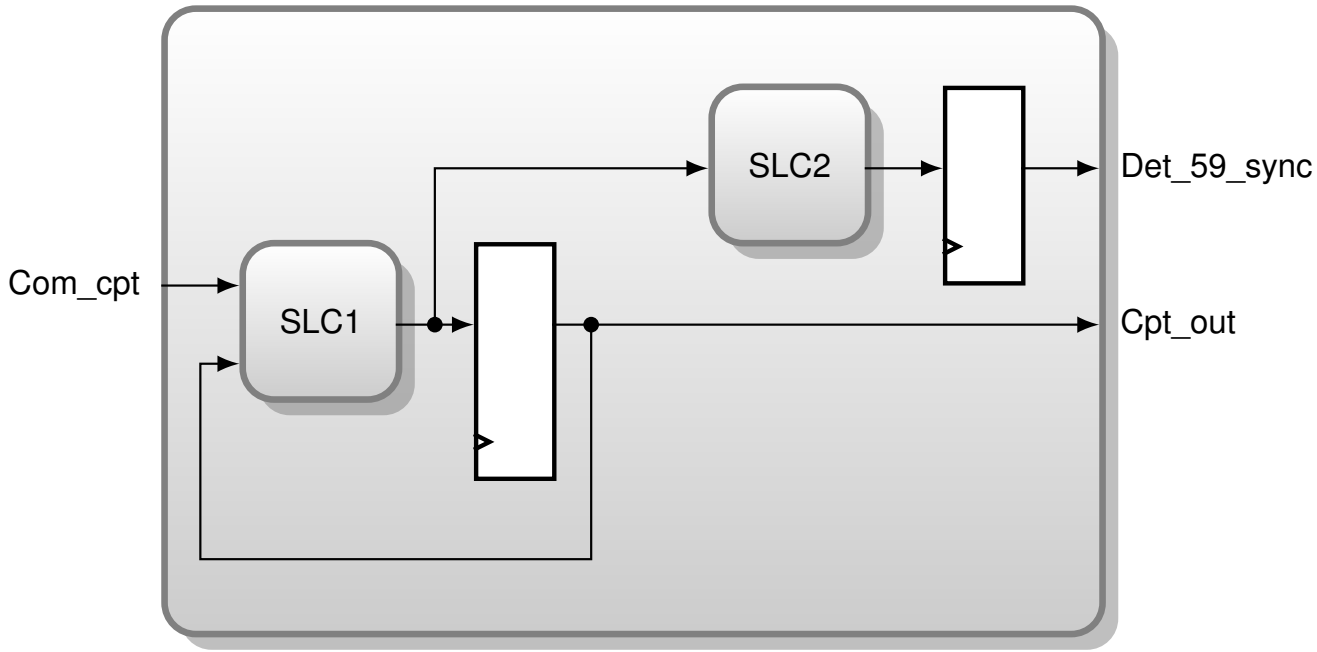
```
Det_59 <= '1' when Cpt_Present = 59 else
          '0';
```

Description utilisant Cpt_Futur et la post-synchronisation

```
process (clk)
begin
  if Rising_Edge(clk) then
    if Cpt_Futur = 59 then
       Det_59_sync <= '1';
    else
       Det_59_sync <= '0';
  end if;
end process;
```

Exemple: report d'un compteur

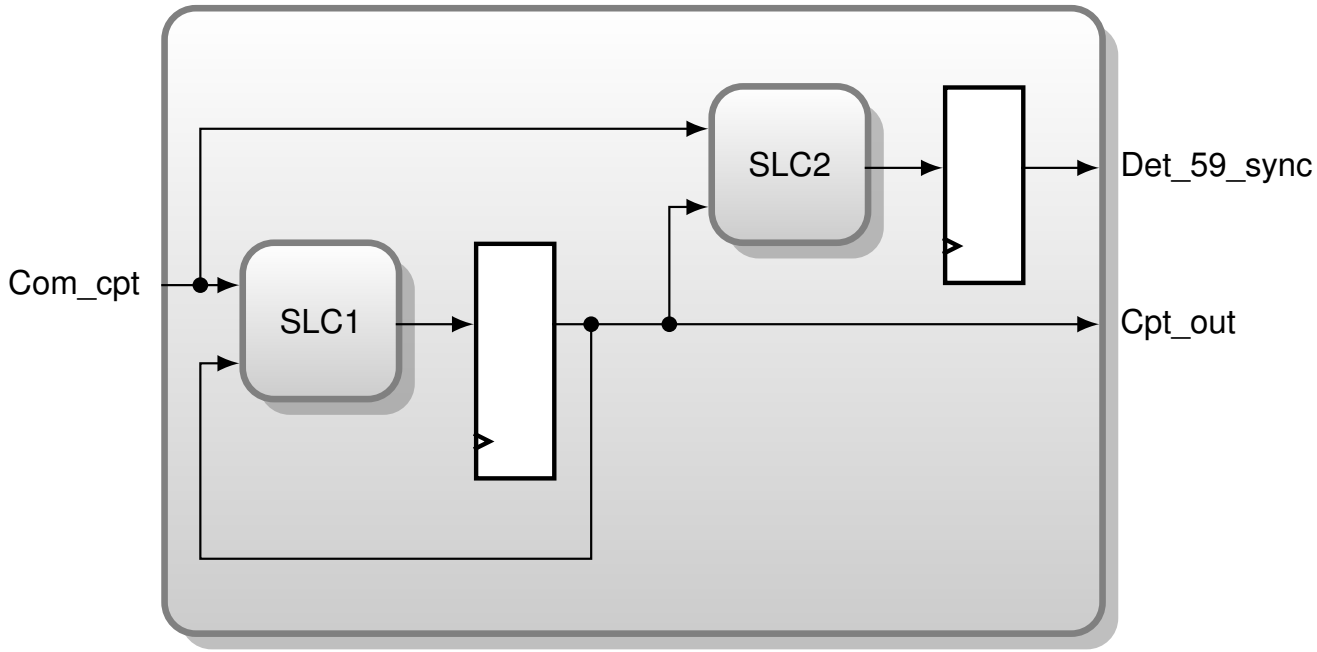
- Cela correspond au schéma suivant:



- Analyse:
  - La sortie Det_59_sync est correcte avec un délai de propagation très court.
  - Par contre la suite des blocs $SLC1 + SLC2$ risque de devenir un nouveau chemin critique

Exemple: report d'un compteur

- Anticiper la détection du compteur:



- Analyse:
  - Solution optimale
  - Attention de bien déterminer la condition "Cond_Comptage" du compteur pour déterminer Det_Fut_59

Datapath-control ou dataflow

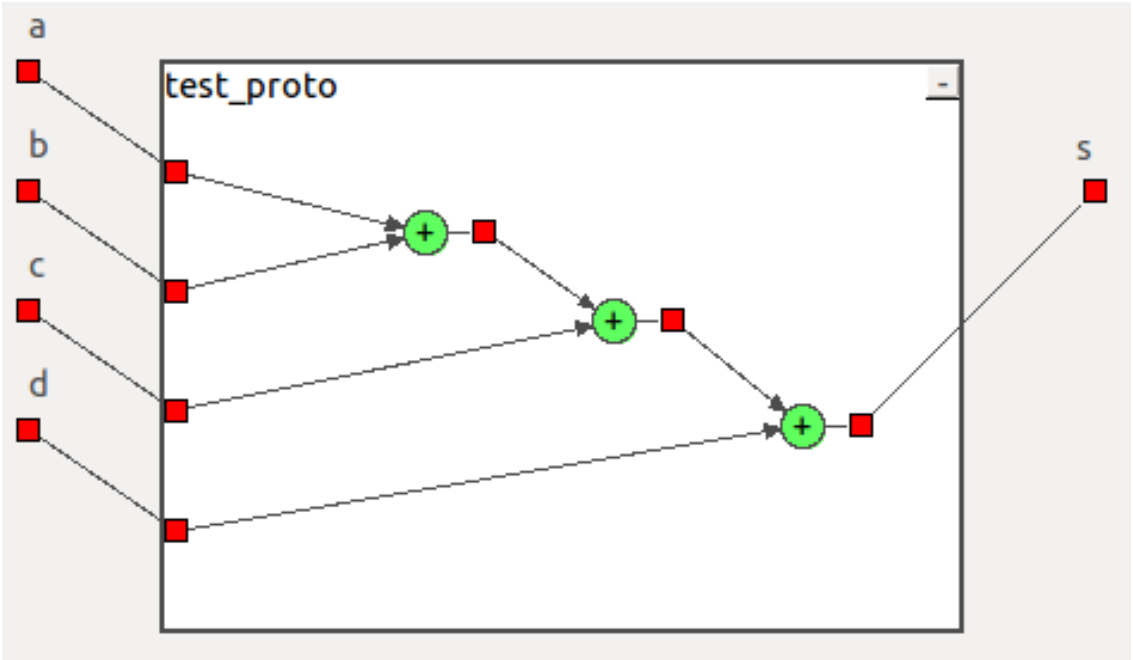
- Flot de données
  - Les données sont transportées avec des signaux de contrôle
  - Pas forcément de contrôle centralisé
  - Exemple:
    - Un signal de validité avec la donnée
    - Un signal de *ready* dans le sens inverse
- Datapath-control
  - Un bloc contient toute la logique nécessaire au traitement
  - Un bloc contient tout ce qui est nécessaire au contrôle

Flot de données

- Exemple: Math2mat `http://www.math2mat.ch`

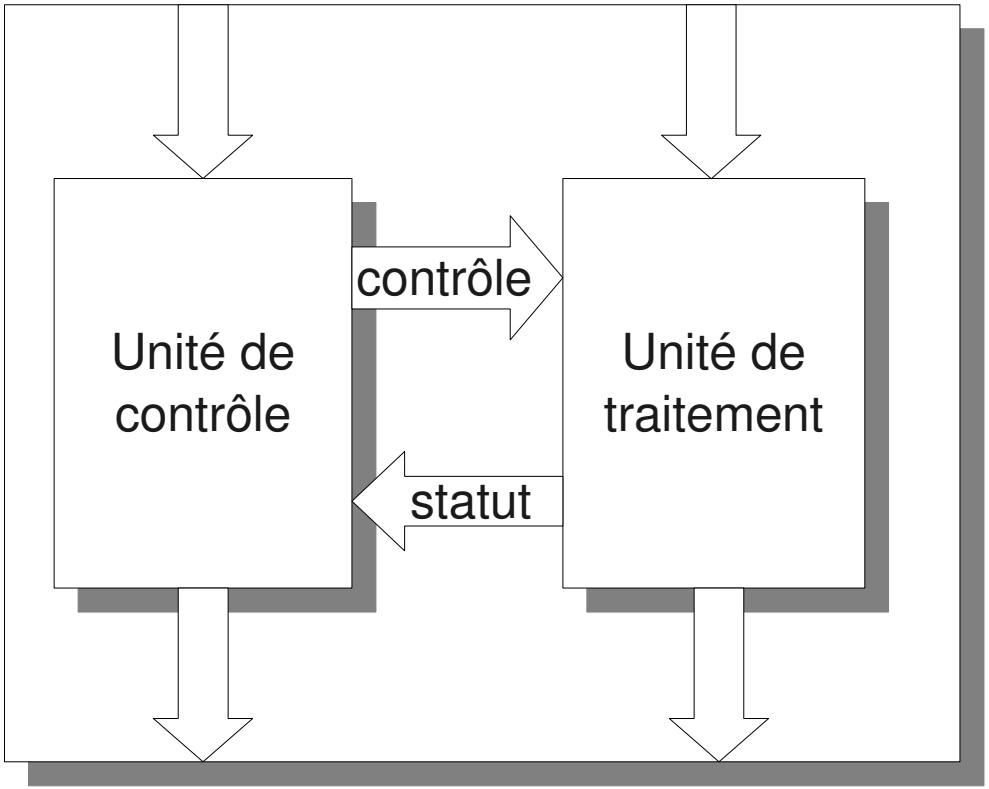
Code matlab

```
function s = test_proto(a,b,c,d)
    s = a + b + c + d;
endfunction
```



Datapath-control

- Séparation de:
  - Traitement de données
  - Contrôle
- Signaux de contrôle (control⇒datapath)
- Signaux de statut (datapath⇒control)



Exemple: report d'un compteur

Description utilisant Cpt_Present et la condition de comptage

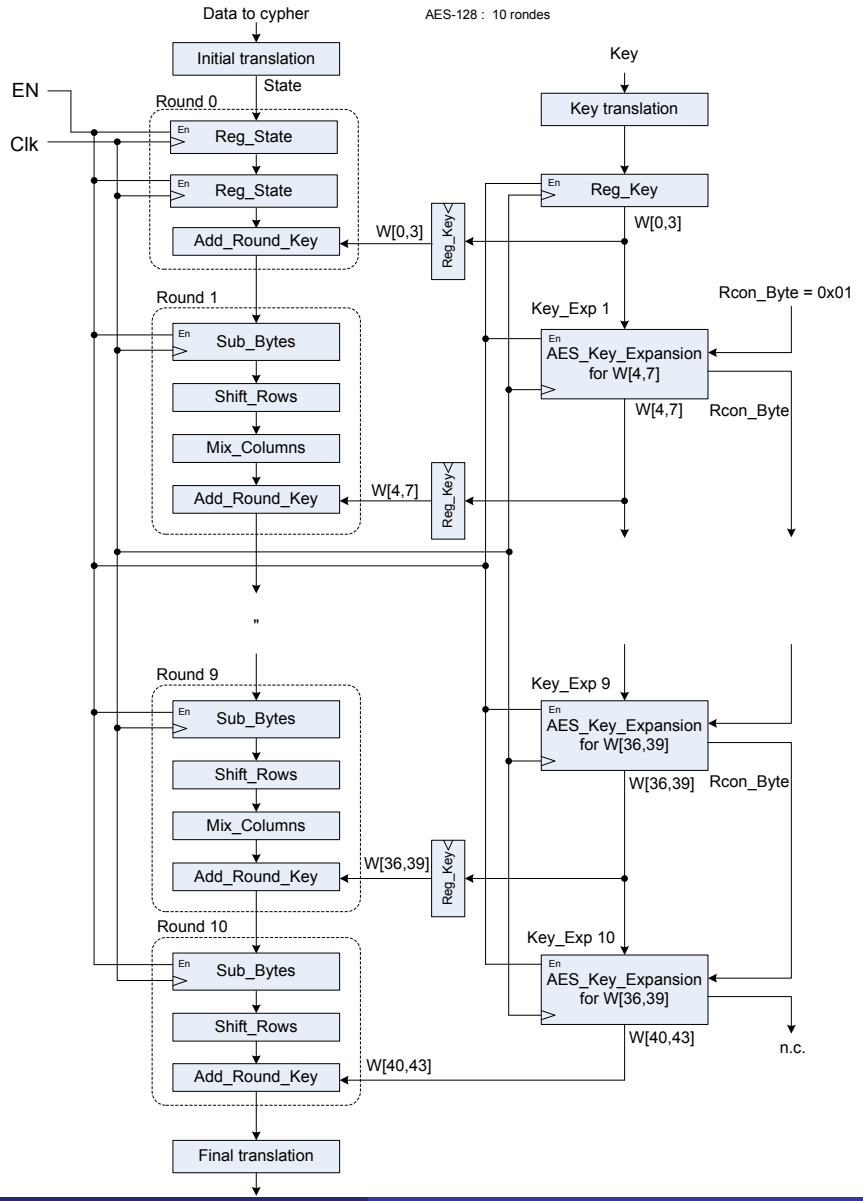
```
Det_59_Fut <= '1' when Cpt_Present = 58 and
                       Cond_Comptage = '1' else
              '0';

process (clk)
begin
  if Rising_Edge(clk) then
       Det_59_sync <= Det_59_Fut ;
  end if;
end process;
```

Exemple: Implantation de AES

- Algorithme de cryptage AES
  - Version 128 bits
  - structure full pipeline :
    - $\Rightarrow$ débit maximum
    - $\Rightarrow$ nécessite beaucoup de logique
  - Structure mixte : séquentiel/pipeline
    - $\Rightarrow$ excellent débit
    - $\Rightarrow$ quantité de logique divisée par deux
      - Remarque: logique divisée par 4 pour FPGA X2C3000 très bon compromis

AES full pipeline

- Implantation V1
  - Vue de la structure de l'implantation "full pipeline" de l'algorithme AES
  - Calcul en parallèle de la clé et du cryptage



Implantation full pipeline

- Description nécessite 200 bloc RAM de 256x8bits
- FPGA XC2V3000 dispose de 96 bloc RAMs de 8Kbits, solde implanté dans des LUT
- Résultat du placement-routage: AES utilise 80% de la FPGA!

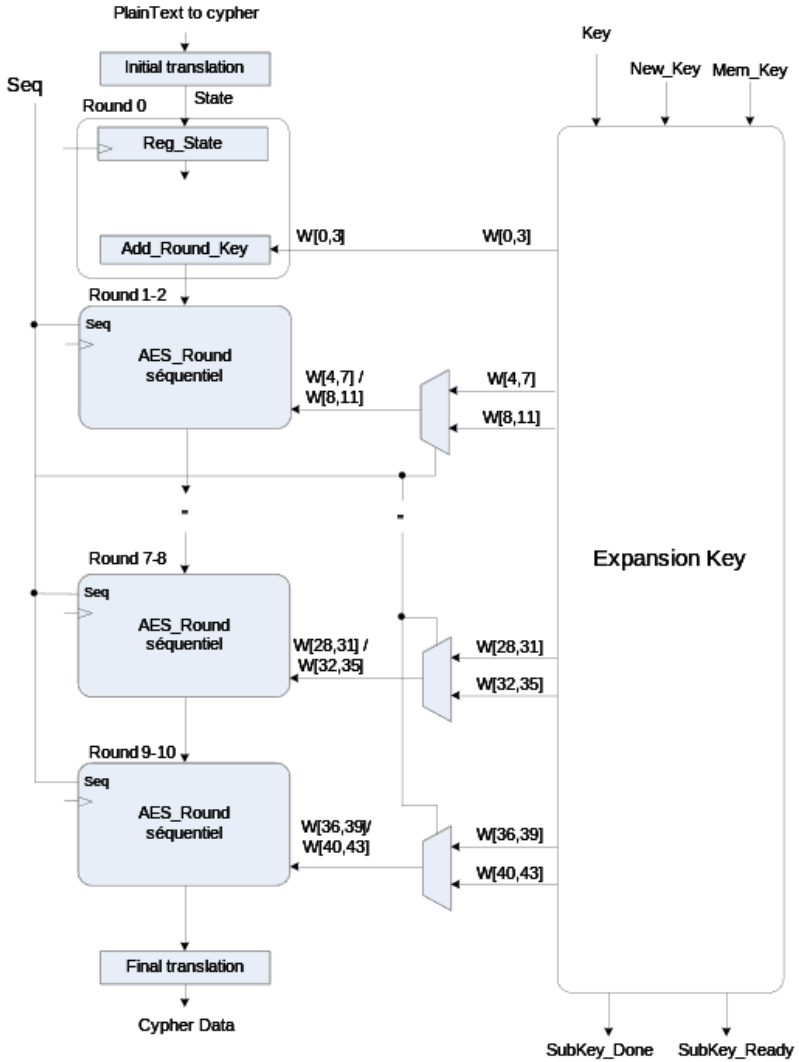
Ressources	**Utilisé**	**Dispo**	**Utilisation**
CLB slices	12'226	14'336	85%
DFFs or latches	3'670	30'220	12%
Block RAMs	96	96	100%

Timing	**Période**	**Fréquence**
System clock	11.464 ns	83.23 MHz

- Débit de l'algorithme AES: 11Gbits/s

AES mixte

- Calcul préalable de la clé en séquentiel, puis mémorisation
- Cryptage en 2 cycles :
  - Un cycle séquentiel
  - Un cycle pipeline
  - Réduction par deux du nombre de rondes



Implantation mixte

- Clé calculée séquentiellement avant le cryptage
- Description nécessite 84 bloc RAM de 256x8bits
- FPGA XC2V3000 dispose de 96 bloc RAMs de 8Kbits, OK
- Résultat du placement-routage:

Ressources	**Utilisé**	**Dispo**	**Utilisation**
CLB slices	2'772	14'336	19%
DFFs or latches	1'595	30'220	5%
Block RAMs	84	96	88%

Timing	**Période**	**Fréquence**
System clock	9.982 ns	100.18 MHz

- Débit de l'algorithme AES: 6Gbits/s

Comparaison des implantations

- Résultat placement-routage des implantations de AES:

Ressources	**Full pipe**	**Seq-pipe**	**Dispo**
CLB slices	12'226 (85%)	2'772 (19%)	14'336
DFFs or latches	3'670 (12%)	1'595 (5%)	30'220
Blocks RAMs	96 (100%)	84 (88%)	96
Timing			
System clock	11.464 ns 87.23 MHz	9.982 ns 100.18 MHz	
Débit AES-128	11 Gbits/s	6 Gbits/s	

- Implantation Seq-Pipe : très bon compromis
  - Débit divisé par 2
  - Quantité de logique divisée par 4 !

Décomposition dans la vie du concepteur

- Le problème de la décomposition est récurrent
- Souvent le développeur commence avec une solution
- Puis doit la revoir et/ou l'affiner
- Très dépendant de la cible
  - FPGA High-end
  - FPGA Low cost