1.0 Principe du pipe-line

Le pipe-line est une technique qui permet d'optimiser l'utilisation des ressources. Supposons une fonction qui calcule $Y = (3A + B)/2$, cela correspond au synoptique de la figure 1 ci-dessous :

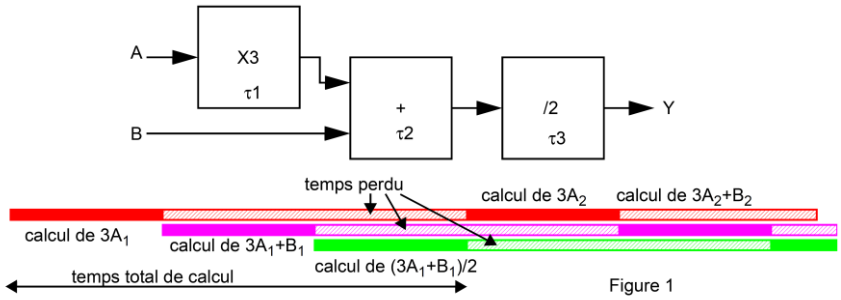


Figure 1

Il apparaît clairement que les différentes fonctions ne sont utilisées que pendant une partie du temps total de calcul, ce qui ne permet pas une utilisation optimale des ressources. Pour pallier ce problème, la solution du pipe-line peut être envisagée, comme le montre le synoptique de la figure 2 ci-dessous. Les blocs Ri sont des registres qui stockent temporairement la donnée appliquée sur leur entrée, tous les registres étant cadencés par une même horloge CLK que l'on supposera dans cet exemple active sur le front montant.

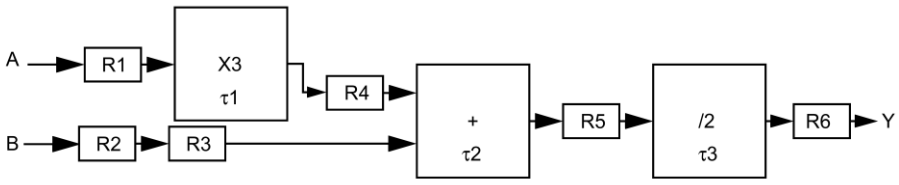
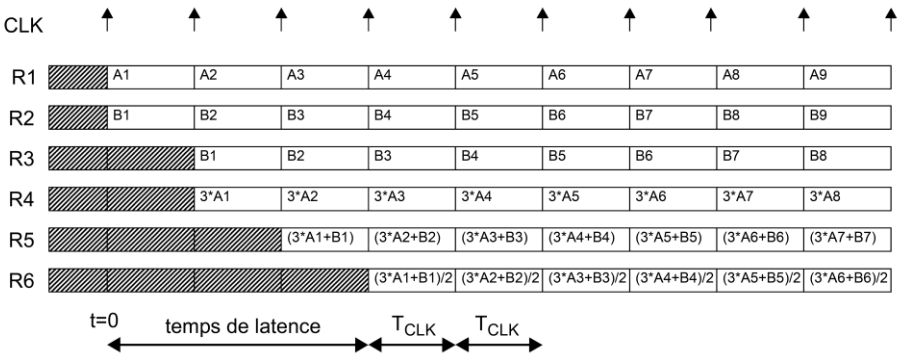


Figure 2

A l'instant $t = 0$, les premières données (A1 et B1) sont appliquées sur les entrées, à chaque front montant d'horloge, de nouvelles données (Ai et Bi) sont appliquées ainsi que le montre la figure 3.



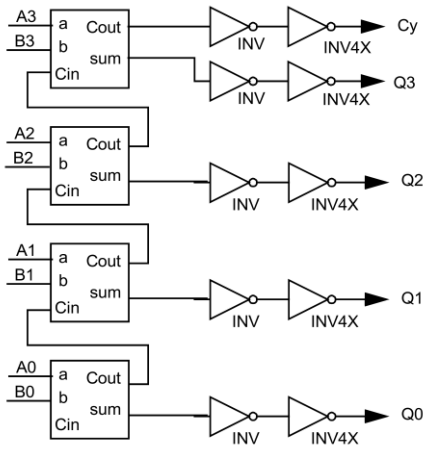
Le cheminement des données dans le circuit n'appelle pas de commentaire particulier, si ce n'est que le premier résultat n'est disponible qu'au bout du quatrième front d'horloge: il existe donc un temps de latence dans le pipe-line, temps qui est directement fonction du nombre de registres présents dans le chemin des données. Par contre, une fois ce temps de latence écoulé, un nouveau résultat est disponible à chaque nouveau front d'horloge, ce qui est plus rapide que la solution de la figure 1, car le délai est de max ($\tau 1$, $\tau 2$, $\tau 3$) au lieu de $\tau 1 + \tau 2 + \tau 3$.

Pour que le pipe-line fonctionne correctement, il faut vérifier que:

- $T_{CLK} \geq \max(\tau 1, \tau 2, \tau 3 \ldots \tau n)$
- Les temps de setup et de hold soient respectés sur les registres
- Les données restent synchronisées tout au long de la chaîne (d'où la présence de R3 dans notre exemple)

2.0 Additionneur pipe-line

Soit le circuit additionneur 4 bits ci-dessous pour lequel les blocs additionneur 1 bit sont des full adders (c.f. cours). Les deux inverseurs de taille croissante permettent de piloter des charges importantes connectées sur les sorties :



Modifier ce schéma pour faire apparaître une structure de type pipe-line. Vous préciserez :

- le type de bascules employé
- les connexions d'horloge
- la séquence des données en entrée et en sortie

3.0 Analyse temporelle

1. Caractérisez l'additionneur 1 bit à l'aide du tableau ci-dessous

- Fan-in sur ses entrées
- Temps de propagation en fonction de la charge connectée sur ses sorties

2. Déterminer la charge vue par les sorties Sum et Cout et déduisez-en le temps de réponse des blocs additionneur dans le circuit. Même question pour les bascules D.

Rappel: pour une porte, le temps de propagation tp se calcule de la façon suivante :

$tp = tp0 + \alpha \sum(fan - in)$. Par exemple, une porte INV qui pilote une porte INV4X aura pour temps de propagation : tp = 0,09 + 0,05 x 3,7 = 0,275 ns

Fonction	Fan-in	tp0 (ns)	α (ns/fan-in)	Pd (μW/MHz)
INV	1	0,09	0,05	0,5
XOR2	1,6	0,3	0,1	0,6
INV4X	3,7	0,08	0,01	2,8
NA2	0,85	0,15	0,1	0,6
NA3	0,9	0,2	0,1	0,6
NA4	1	0,25	0,1	0,6
NO2	0,75	0,15	0,1	0,6
NO3	0,8	0,2	0,15	0,6
OR2	0.9	0.3	0.06	0.9
AN2	0,9	0,3	0,06	0,9

Fonction	Fan-in (D,J ou K)	Fan-in CLK	tp0 (ns) CLK to Q	tp0 (ns) D to Q	α (ns/fan-in)	t_{SU} (ns)	t_H (ns)
D flip-flop	0,9	1	1,2		0,06	0,4	0,5
JK flip-flop	0,7	0,9	1,6		0,06	0,5	0,02
D latch	0,70.	0,9	0,5	0,7	0,05		

3. Déterminer le temps de latence de l'additionneur.

4. Déterminer la fréquence maximale d'horloge. En déduire le temps de réponse de l'additionneur. Comparez avec le résultat obtenu pour le circuit équivalent à propagation de retenue.

5. Quel serait le temps de latence et de réponse pour une extension à 16 bits? Comparez avec le résultat que l'on obtiendrait pour le circuit équivalent à propagation de retenue..