Question 1 (5 points) Architectures scalaires, déroulement de boucles et superscalaires.

Soit un pipeline DLX de type M4 à cinq (5) niveaux et soit la boucle suivante (Figure 1.1) qui réalise l'opération vectorielle Z[i] = (A[i] + B[i] + C[i]) * (D[i] + E[i]) * F[i] où Z, A, B, C, D, E et F sont des vecteurs de dimension 100 de mots doubles. Les éléments des vecteurs A, B, C, D, E et F sont chargés dans F10, F20, F30, F40, F50 et F60 respectivement, à partir des registres R1, R2, R3, R4, R5 et R6 qui eux sont initialisés respectivement aux adresses 800, 1600, 2400 et 3200, 4000 et 4800. Finalement, la valeur est retournée dans le vecteur Z à partir de l'adresse contenue dans R6 qui est de 5600.

```
L : LD      F10, 0(R1)
    LD      F20, 0(R2)
    ADDD    F70, F10, F20
    LD      F30, 0(R3)
    ADDD    F70, F70, F30
    LD      F40, 0(R4)
    LD      F50, 0(R5)
    ADDD    F80, F40, F50
    LD      F60, 0(R6)
    MULTD   F80, F80, F70
    MULTD   F80, F80, F60
    SD      0(R7), F80
    MSUBI   R1, R7, #8      // Cette instruction a été créer pour les besoins du problème et éviter d'avoir
                            // à écrire 7 décrémentations consécutives et ainsi débordé sur la figure 1.1
                            // Pour les besoins du problème supposer donc que l'instruction se fait en 1 cycle de EX
    BNEQZ   R1, L
```

a) (2 pts) Complétez la figure 1.1. Vous devez tenir compte des cycles de suspension ou d'attente. Considérez un modèle M4. Finalement, donnez le nombre de cycles pour 100 itérations.

b) (2 pts) À partir du résultat de la figure 1.1, complétez la figure 1.2. Pour cela, optimisez le code en a) en déroulant au besoin la boucle et en réorganisant au besoin le code, tout en faisant disparaître du même coup le maximum de suspensions. Expliquez bien votre démarche. Calculez l'accélération.

c) (1 pt) Soit une architecture superscalaire avec 1 pipeline pour le calcul entier et 1 pipeline pour le calcul flottant, ce qui veut dire être capable de lire 2 instructions et lire/écrire 2 données en parallèle. À partir des résultats de la figure 1.2, faites une approximation du nombre de cycles nécessaires pour 100 itérations de la boucle. Expliquez bien votre démarche. Estimez l'accélération.

*En analysant la figure en Annexe, on s'aperçoit qu'il faut considérer le plus long chemin entre les 2 pipelines. Comme on voit dans la figure 1.2 qu'il y a plus de LD i.e. 28, cette dernière plus MSUBI et BNEQZ devrait donné une bonne approximation. Donc une trentaine d'instruction. Ce qui donne 30 cycles * 25 = 750 cycles donc un gain de 2600/750 = 3.4.*

##		Instructions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	2	22	23	24	25	26	27	28	29
1	L:	LD F10, 0(R1)	*LI*	*DI*	*EX*	*ME*	*ER*																								
2		LD F20, 0(R2)		*LI*	*DI*	*EX*	*ME*	*ER*																								
3		ADDD F70, F10, F20			*LI*	*DI*	*SU*	*E1*	*E2*	*E3*	*ME*	*ER*																			
4		LD F30, 0(R3)				*LI*	*SU*	*DI*	*EX*	*ME*	*ER*																				
5		ADDD F70, F70, F30						*LI*	*DI*	*SU*	*E1*	*E2*	*E3*	*ME*	*ER*																
6		LD F40, 0(R4)							*LI*	*SU*	*DI*	*EX*	*ME*	*ER*																	
7		LD F50, 0(R5)									*LI*	*DI*	*EX*	*SU*	*ME*	*ER*															
8		ADDD F80, F40, F50										*LI*	*DI*	*SU*	*SU*	*E1*	*E2*	*E3*	*ME*	*ER*											
9		LD F60, 0(R6)											*LI*	*SU*	*SU*	*DI*	*EX*	*ME*	*ER*												
10		MULTD F80, F80, F70														*LI*	*DI*	*SU*	*E1*	*E2*	*E3*	*E4*									
11		MULTD F80, F80, F60															*LI*	*SU*	*DI*	*SU*	*SU*	*SU*	*E1*	*E2*	*E3*	*E4*	*ME*	*ER*			
12		SD 0(R7), F80																	*LI*	*SU*	*SU*	*SU*	*DI*	*EX*	*SU*	*SU*	*SU*	*ME*			
13		MSUBI R1, R7, #8																					*LI*	*DI*	*SU*	*SU*	*SU*	*EX*	*ME*	*ER*	
14		BNEQZ R1, L																						*LI*	*SU*	*SU*	*SU*	*DI*	*EX*		
15																												*LI*	*SU*	*LI*	
16																															

Figure 1.1 À compléter pour la question 1a)

27 cycles x 100 = 2700 cycles

Cycle	Pipeline 1		Pipeline 1 (suite)
1	*LD F10, 0(R1)*	35	*LD F62, -16(R6)*
2	*LD F11, -8(R1)*	36	*LD F63, -24(R6)*
3	*LD F12, -16(R1)*	37	*MULTD F80, F80, F70*
4	*LD F13, -24(R1)*	38	*MULTD F81, F81, F71*
5	*LD F20, 0(R1)*	39	*MULTD F82, F82, F72*
6	*LD F21, -8(R1)*	40	*MULTD F83, F83, F73*
7	*LD F22, -16(R1)*	41	*MULTD F80, F80, F60*
8	*LD F23, -24(R1)*	42	*MULTD F81, F81, F61*
9	*ADDD F70, F10, F20*	43	*MULTD F82, F82, F62*
10	*ADDD F71, F11, F21*	44	*MULTD F83, F83, F63*
11	*ADDD F72, F12, F21*	45	*SD 0(R7), F80*
12	*ADDD F73, F13, F23*	46	*SD -8(R7), F81*
13	*LD F30, 0(R3)*	47	*MSUBI R1, R7, #32*
14	*LD F31, -8(R3)*	48	*BNEQZ R1, L*
15	*LD F32, -16(R3)*	49	*SD 16(R7), F82*
16	*LD F33, -24(R3)*	50	*SD 8(R7), F83*
17	*ADDD F70, F70, F30*	51	
18	*ADDD F71, F71, F31*	52	
19	*ADDD F72, F72, F32*	53	
20	*ADDD F73, F73, F33*	54	
21	*LD F40, 0(R4)*	55	
22	*LD F41, -8(R4)*	56	
23	*LD F42, -16(R4)*	57	
24	*LD F43, -24(R4)*	58	
25	*LD F50, 0(R5)*	59	
26	*LD F51, -8(R5)*	60	
27	*LD F52, -16(R5)*	61	
28	*LD F53, -24(R5)*	62	
29	*ADDD F80, F40, F50*	63	
30	*ADDD F81, F41, F41*	64	
31	*ADDD F82, F42, F41*	65	
32	*ADDD F83, F43, F43*	66	
33	*LD F60, 0(R6)*	67	
34	*LD F61, -8(R6)*	68	

Tableau 1.2 À compléter pour la question 1b)

50 cycles * 25 = 1250 cycles donc un gain de 2600/1250 = 2.08

Question 2 (3 points) Prédiction de branchement

a) (.75 pt) Expliquez ce que représente un aléa de contrôle sur une architecture RISC, ainsi que les implications négatives de ce type d'aléas sur l'architecture RISC.

Expliquez à la page 18 du bloc 1 chap. 3

b) (.75 pt) Donnez un exemple de branchement pour lequel le branchement statique avec anticipation peut être acceptable.

Si on test un 1 bit sur 2^{32}…

c) (.75 pt) Soit la boucle de la figure 1.1 du no 1 pour laquelle on implémente une stratégie d'anticipation dynamique avec un compteur de 1 bit associé à chaque branchement. À quelle fréquence va-t-on faire une mauvaise anticipation et donc être forcé de vider le pipeline. Expliquez.

On va se tromper 2 fois sur 100 i.e. en entrant et en sortant

d) (.75 pt) Soit la boucle de la figure 1.1 du no 1 pour laquelle on implémente une stratégie d'anticipation dynamique avec un compteur de 2 bits associé à chaque branchement. À quelle fréquence va-t-on faire une mauvaise anticipation et donc être forcé de vider le pipeline. Expliquez.

On va se tromper 1 fois sur 100 i.e. en entrant à la 2^e itération mais pas en sortant.

Question 3 (2 points) Synthèse de haut niveau

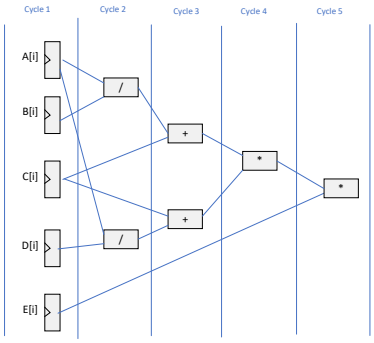
Soit l'expression Z[i] = (A[i] / B[i] + C[i]) * (A[i] / D[i] + C[i]) * E[i] qu'on désire implémenter en matériel avec Vivado HLS dans une boucle *for* dans laquelle on insère dans un pragma *HLS PIPELINE*.

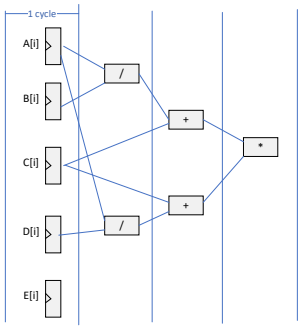
a) (1 pt) Complétez la figure 3.1 afin de démontrer comment le pipeline se remplit au cours des 5 premiers cycles de l'horloge (la lecture des données constituant le 1er étage du pipeline). Assumez qu'il n'y a pas de contraintes sur les resources.

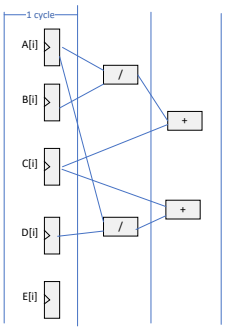
b) (1 pt) En analysant votre résultat de a), montrez une manière de déterminer le nombre d'instance(s) pour chaque type d'opération (/, + et *) afin d'assurer un débit (II) de 1. Donnez ce nombre d'instances pour chaque type d'opération.

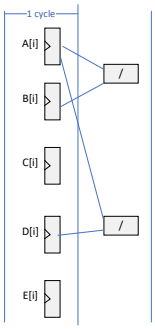
On peut voir lequel des 4 cycles prend le maximum d'instances pour chaque type. Le cycle 4 demande quand le pipeline est plein demande 1 additionneur, 2 multiplieurs et 2 diviseurs.

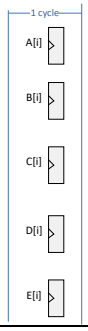
Utilisez les symboles suivants :


Cycle 1
Cycle 2
Cycle 3
Cycle 4
Cycle 5
A[i]
B[i]
C[i]
D[i]
E[i]
/
+
*


1 cycle
A[i]
B[i]
C[i]
D[i]
E[i]
/
+
*


1 cycle
A[i]
B[i]
C[i]
D[i]
E[i]
/
+


1 cycle
A[i]
B[i]
C[i]
D[i]
E[i]
/


1 cycle
A[i]
B[i]
C[i]
D[i]
E[i]

Question 4 (3 points) Laboratoire no 2

a) (1 pt) Soit un calcul de multiplication matricielle 60x60 auquel on a appliqué le pragma *hls pipeline* entre L2 et L3 et l'utilisation du pragma *array_partition complete* sur les matrices *a* (DIM = 2) et *b* (DIM=1). Donnez la valeur de x, y, z, u et v qui complète les 5 affirmations suivantes (supposez qu'on a des mémoires dual port):

- pour un débit de 1, Vivado HLS va organiser les entrées en 1 fois x lectures
- pour un débit de 2, Vivado HLS va organiser les entrées en 2 fois y lectures.
- pour un débit de 3, Vivado HLS va organiser les entrées en 3 fois z lectures.
- pour un débit de 4, Vivado HLS va organiser les entrées en 4 fois u lectures.
- pour un débit de 5, Vivado HLS va organiser les entrées en 5 fois v lectures.

x= 30, y = 15, z = 10, u = 8 et v = 6

b) (1 pt) Pour un calcul de multiplication matricielle 60 x 60 en *int* pour lequel on a un pragma *hls pipeline* entre L2 et L3, combien de ressources DSP48E sont-elles requises (assumez 3 instances/DSP). On assume aussi que le *array_partition* a été spécifié pour assurer II=1. Expliquez votre réponse.

*Ici on aura O(N) en ressources implique 60 * 3 instances soit 180 instances.*

c) (1 pt) Pour un calcul de multiplication matricielle 15 x 15 en *int* pour lequel on a pragma *hls pipeline* entre L1 et L2, combien de ressources DSP48E sont-elles requises (assumez 3 instances/DSP). On assume aussi que le *array_partition* a été spécifié pour assurer II=1. Expliquez votre réponse.

*Ici on aura $O(N^2)$ en ressources implique 15x15 * 3 = 675 instances.*

Question 5 (3 points) Laboratoire no 1 partie 3

a) (1 pt) Expliquez la différence entre le mode SMP (Symmetric Multi-Processing) et le mode AMP (Asymmetric Multi-Processing) et indiquez avec lequel des 2 modes vous avez travaillé? Justifiez.

SMP est 1 OS pour tous les cœurs. AMP implique 1 OS (ou pas) par cœur. Nous on avait 1 uC/OS-III / cœur donc on est AMP. Cœur

b) (1 pt) Quels sont les avantages/désavantages de l'approche utilisée soit la DDR partagée (Fig. 5.1) combinée au *handshacking* (poignée de main)?

```
volatile uint32_t* req = (uint32_t*) (BASEADDR + 0x04); // signal comme quoi on est prêt à recevoir un data
volatile uint32_t* ack = (uint32_t*) (BASEADDR + 0x00); // signal comme quoi on attend que le producteur soit pret
volatile uint32_t* done = (uint32_t*) (BASEADDR + 0x08);
volatile uint32_t* burst_no = (uint32_t*) (BASEADDR + 0x0C); // No du burst
volatile uint32_t* number_of_packets = (uint32_t*) (BASEADDR + 0x10); // Nombre de paquets
```

Figure 5.1

Tel qu'expliqué au lab il s'agit d'approche simple sans besoin d'IP de Vivado. Toutefois le besoin de se synchroniser continuellement amène une perte d'efficacité i.e. l'attente active…

c) On désire faire la communication entre core0 et core1 via un FIFO traditionnel plutôt que par *handshacking* (poignée de main). Comment pourrait-on réaliser un tel FIFO à partir de notre zone de mémoire partagée? Expliquez.

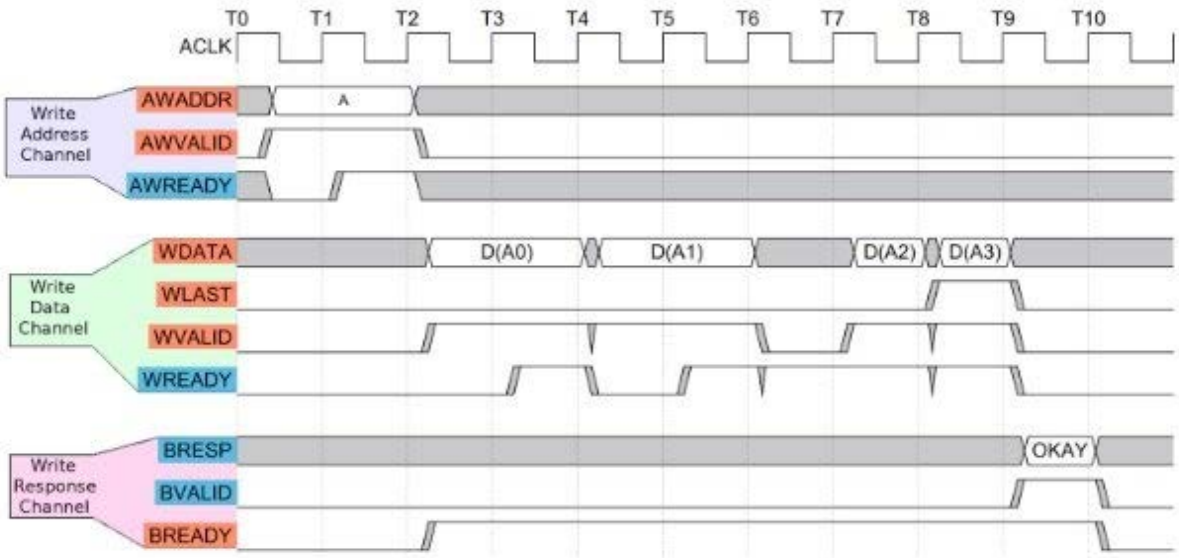
On pourrait utiliser un mutex matériel de la libraire Vivado. Donc le pointeur de tête et de fin et l nombre d'éléments seraient gérer par le mutex hardware avec accès bloquant. Autre possibilité serait d'utiliser le mailbox de Vivado ou à chaque écriture lecture le consammateur/producteur peut recevoir une interruption.

Question 6 (3 points) Interconnexions AXI

a) (1 pt) Présentez et expliquez le rôle des principaux acteurs d'une transaction AXI4.

Ceux-ci sont clairement présenté dans le bloc 3 du chapitre.

b) (2 pts) Complétez le schéma de la figure 6.1 pour une écriture de 4 mots entre 1 maître et 1 esclave à partir d'une adresse A.



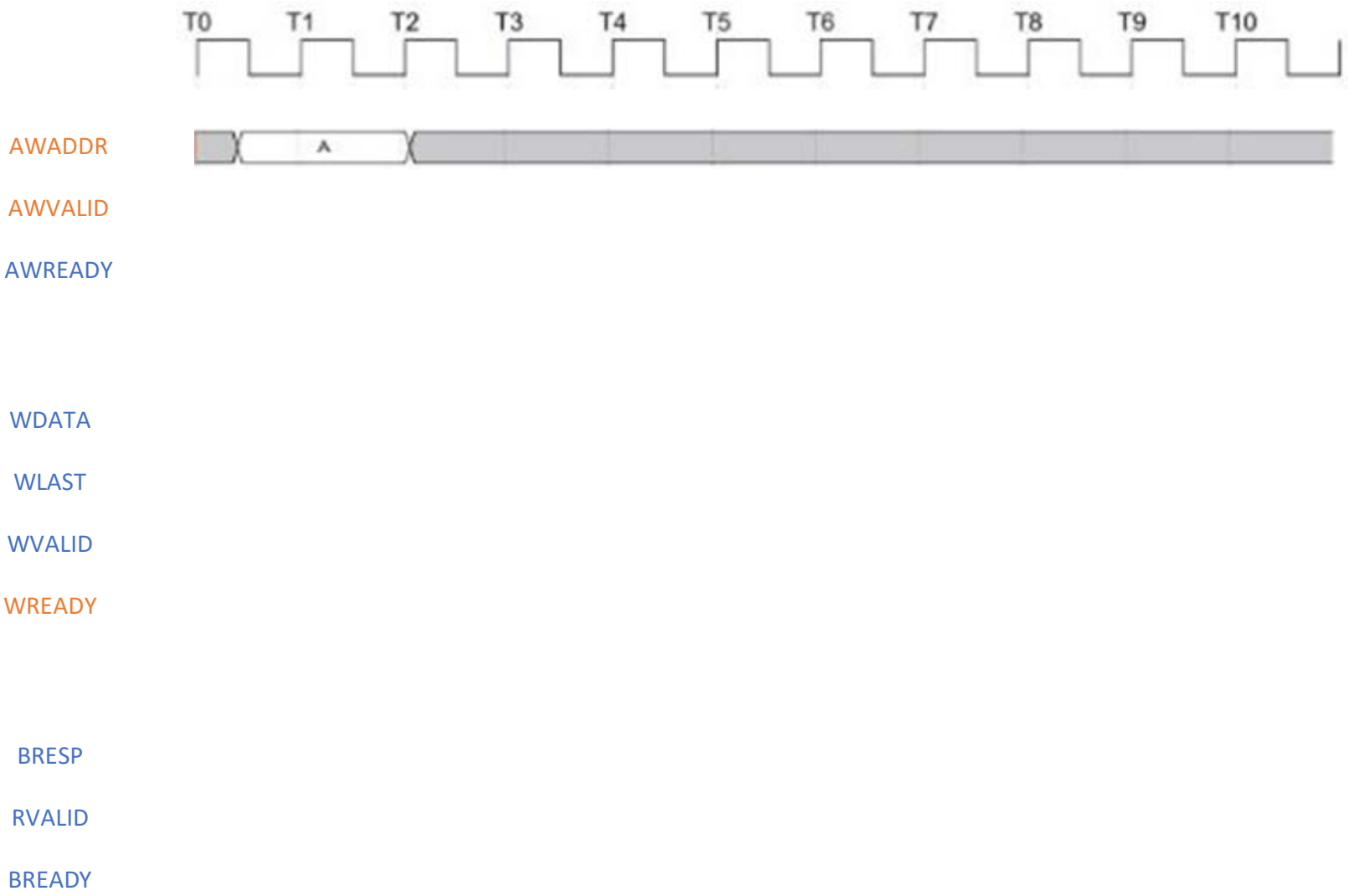


Figure 6.1 À compléter pour le no 6b (n.b. en orange ce sont les signaux émis du maitre et en bleu de l'esclave)

Question 7 (1 point) Blocage

Soit une application composée de 4 tâches Task_1 à Task_4 en ordre décroissant de priorité et utilisant quatre mutex M1, M2 et M3:

	M1	M2	M3
Task_1	8	0	3
Task_2	0	1	7
Task_3	0	0	6
Task_4	9	10	8

a) (.5 pt) Donnez le blocage de B1, B2, B3 et B4 pour héritage de priorité. Détaillez vos calculs.

B1 = max(9)+max(6, 7, 8) = 17
B2 = max(9)+max(10)+max(6,8) = 26
B3 = max(9)+max(10)+max(8) = 27
B4 = 0

b) (.5 pt) Donnez le blocage de B1, B2, B3 et B4 pour ICPP. Détaillez vos calculs.

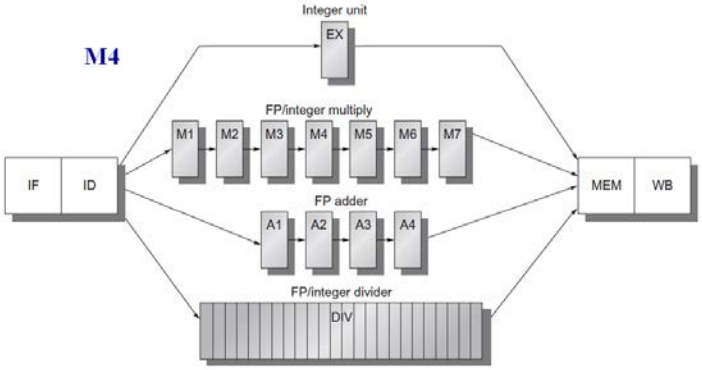
B1 = max(max(9), max(6, 7, 8)) = 9
B2 = max(max(9), max(10), max(6,8)) = 10
B3 = max(max(9), max(10), max(8)) = 10
B4 = 0

Annexe

Détails des instructions pouvant être pipelinées	Nom de l'instruction	Nombre de cycles dans EX	Cycle du pipeline où l'opération termine
MOV R1, R5	**Copie R 1 dans R2**	**1**	**ER (le résultat est dans l'accumulateur après EX)**
LD F1, 0(R1)	**À partir de l'adresse contenue dans R1 auquel on additionne 0, chargement du double mot dans F1**	**1**	**ER (le résultat est dans l'accumulateur après MEM)**
ADDI R1, R1, #8	**Addition immédiate : R1 –> R1 + 8**	**1**	**ER (le résultat est mis dans l'accumulateur après EX)**
SLLI R8, #3	**Décalage à gauche de 3**	**1**	**ER (le résultat est dans l'accumulateur après EX)**
ADD R1, R1, R3	**Addition de deux mots : R1 –> R1 + R3**	**1**	**ER (le résultat est mis dans l'accumulateur après EX)**
MULT R8,R5,R1	**Multiplie 2 mots (32 bits) R8 <- R8 X R7**	**2**	**ER (le résultat est dans l'accumulateur après EX)**
ADDD F1, F1, F3	**Addition de deux doubles mots : F1 –> F1 + F3**	**3**	**ER (le résultat est mis dans l'accumulateur après E3)**
MULTD F1, F1, F5	**Multiplication de deux doubles mots : F1 –> F1 * F5**	**4**	**ER (le résultat est mis dans l'accumulateur après E6)**
SD 0(R2), F6	**Rangement d'un mot à partir de F6**	**1**	**MEM**
BNEQ R3, etiq	**Branch si non nul**	**1**	**EX**
MSUBI R1, R7, #8 **N.B. Ce genre d'instruction n'existe probablement pas, c'est simplement ici pour simplifier le code et ne pas avoir à écrire 7 décrémentations consécutives.**	**Soustraction immédiate multiple pour les registres allant de R1 à R7 : R1 –> R1 – 8, …, R7 –> R7 - 8**	**1**	**ER**

Figure A1. Détail des instructions du DLX pour no 1.
Considérez qu'il s'agit d'un modèle M4 et qu'on a un seul port de mémoire aux étapes de LI et de ME du pipeline

Autres modèles possibles:
•Modèle pipeline étendu aux opérations flottantes.



Architecture superscalaire

PIPELINE ENTIER

```
B:
LD      F0,0(R1)
LD      F6,-8(R1)
LD      F10,-16(R1)
LD      F14,-24(R1)
LD      F18,-32(R1)
SD      0(R1),F4
SD      -8(R1),F8
SD      -16(R1),F12
SUBI    R1, R1, #40
BNEZ    R1, B
SD      16(R1),F16
SD      8(R1), F20
```

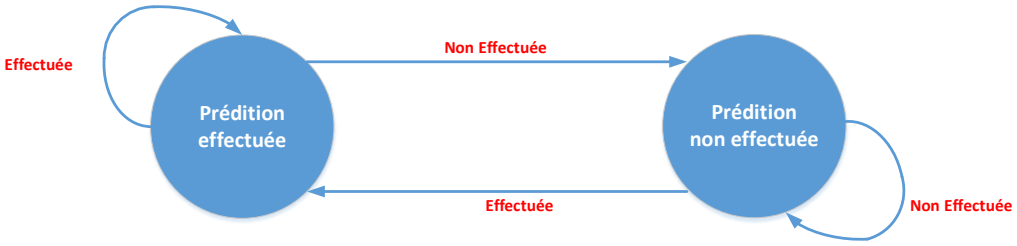
1 SU

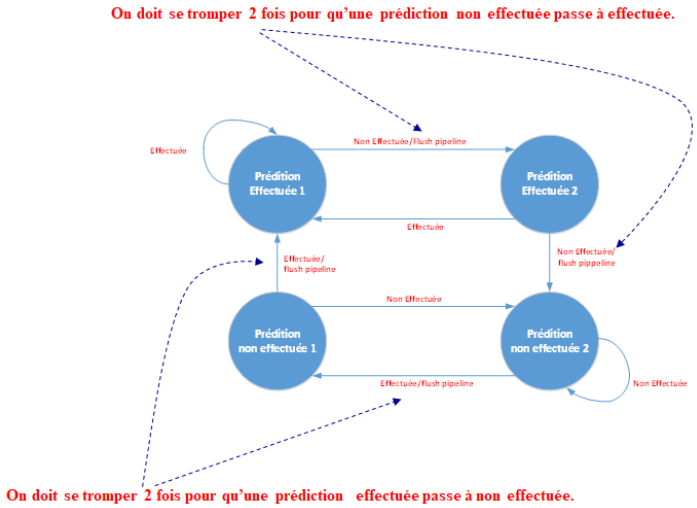
2 SU

PIPELINE FLOTTANT

```
B:

ADDD    F4,F0,F2
ADDD    F8,F6,F2
ADDD    F12,F10,F2
ADDD    F16,F14,F2
ADDD    F20,F18,F2
```

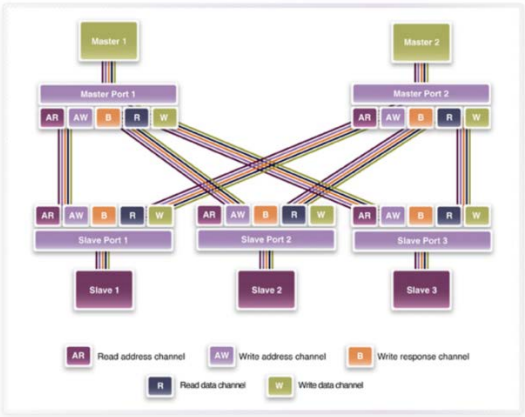
```
  1 itération  =    12 cycles
200 itérations =  2400 cycles

A = 10000/2400  = 4,17
```





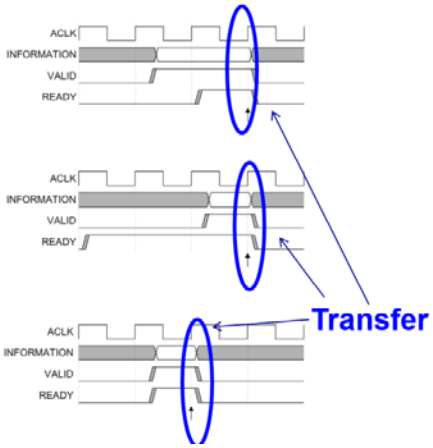
AMBA AXI

Exemple d'un multilayer (2 maitres et 3 esclaves):



Flôt de contrôle

- Le transfert a lieu seulement quand:
  - La source est Valid, et
  - la destination est Ready
- Sur chaque canal, le maître et l'esclave peuvent limités le flôt.
- Très flexible



Formules du blocage:

- **Avec l'héritage de priorité on a:**

$$B_i = \sum_{k=1}^{K} usage(k,i)CS(k)$$

Où:

1. ***usage*** **est une fonction 0/1: *usage(k,i)* = 1 si la ressource *k* est utilisé par au moins une tâche de priorité inférieure à *i*, et au moins une tâche de priorité supérieure ou égale à *i*, sinon 0.**
2. **Lorsque qu'on a plusieurs usage(k,i) = 1 pour un même k, on prend le maximum**
3. **CS(*k*) le temps maximum requis pour passer au travers la section critique *k*.**
4. ***K*** **étant le nombre total de ressources (mutex)**

- **Avec ICPP on a:**

$$B_i = \max_{k=1}^{K} usage(k,i)CS(k)$$

Où:

1. **usage est une fonction 0/1: usage(k,1) = 1 si la ressource k est utilisé par au moins une tâche de priorité inférieure à i, et au moins une tâche de priorité supérieure ou égale à i, sinon 0.**
2. **Lorsque qu'on a plusieurs usage(k,i) = 1 pour un même k, on prend le maximum**
3. **CS(k) le temps maximum requis pour passer au travers la section critique k.**
4. **K étant le nombre total de ressources (mutex)**