

Chapitre 4

La division

4.1 Présentation

4.1.1 Principe de la division

L'opération de division ne conduit pas à un opérateur arithmétique structuré cascadable ou arborescent comme l'addition, la soustraction ou la multiplication. Il s'agit d'une opération plus complexe reposant sur les autres opérateurs. Toutefois, comme pour les autres opérateurs, le principe d'implémentation de la division s'inspire de la manière de diviser 2 nombres à la main. Un exemple est illustré en figure 4.1. En base 10, le diviseur multiplié par des puissances de 10 est soustrait au dividende autant de fois que possible. En base 2, les multiplications sont avantageusement remplacées par des décalages à gauche. L'implémentation de la division reprend cette démarche. Elle peut être réalisée de façon logicielle (soft) sur une UAL donnée, ou matérielle (hard). Une réalisation hard présente l'avantage d'être optimisée et donc plus rapide. Ce TD/TP a pour objectif ce type de réalisation hard de diviseur.

$$\begin{array}{r|l} 156 & 12 \\ 120 & 13 \\ \hline 36 & \\ 36 & \\ \hline 0 & \end{array}$$

FIGURE 4.1 – Exemple de division en base 10. $156=12*10+3*12$

4.1.2 Algorithme de la division

De nombreux algorithmes existent pour implémenter une division de nombres binaires. L'algorithme choisi dans ce TD/TP est l'algorithme sans restauration (Non Restoring Division Algorithm).

Les étapes de cet algorithme sont décrites ci-dessous, et un exemple suit dans le tableau 4.2.

Algorithme de division sans restauration :

- Décalage des registres P et A de 1 bit vers la gauche.
- Si le contenu du registre P est :

- négatif : on additionne le contenu du registre B à P ($P+B$)
- positif : on soustrait le contenu du registre B à P ($P+(-B)$)
- Si le résultat chargé dans P est négatif, on met le bit de poids faible de A (A_0) à « 0 » sinon à « 1 ».
- Ces étapes sont répétées N fois, N étant le format des données, ici $N=4$.

Opération élémentaire	P $P_3P_2P_1P_0$	A $A_3A_2A_1A_0$	Commentaires
Chargement des registres	00000	0111	Initialisation des registres. Chargement du dividende (A) et du diviseur (B)
décalage	00000	111-	décalage P,A de 1 bit à gauche
-B	11110	111-	$P \geq 0$, on soustrait le diviseur B ($P+(-B)$).
$A_0 \leq 0$	11110	111 <u>0</u>	résultat < 0 , A_0 mis à « 0 ».
décalage	11101	11 <u>0</u> -	décalage P,A de 1 bit à gauche
+B	00010	11 <u>0</u> -	$P < 0$, on additionne le diviseur B ($P+B$).
$A_0 \leq 0$	11111	11 <u>00</u>	résultat < 0 , A_0 mis à « 0 ».
décalage	11111	1 <u>00</u> -	décalage P,A de 1 bit à gauche
+B	00010	1 <u>00</u> -	$P < 0$, on additionne le diviseur B ($P+B$).
$A_0 \leq 1$	00001	1 <u>001</u>	résultat > 0 , A_0 mis à « 1 ».
décalage	00011	<u>001</u> -	décalage P,A de 1 bit à gauche
-B	11110	<u>001</u> -	$P \geq 0$, on soustrait le diviseur B ($P+(-B)$).
$A_0 \leq 1$	00001	<u>0011</u>	résultat > 0 , A_0 mis à « 1 ».
		quotient	

FIGURE 4.2 – Algorithme de la division sans restauration. Exemple de la division de 7 (0111) dans le registre A par 2 (0010) dans le registre B. Remarque : la soustraction de 2 est représentée dans le tableau par l'addition de -2 (11110). Le quotient obtenu 0011 (3) correspond au résultat de la division de 7 par 2.

4.2 Réalisation matérielle

4.2.1 Schéma de principe

L'implémentation matérielle de l'algorithme de division nécessite une structure logique avec une unité de traitement (UT) dédiée aux opérations élémentaires, et une unité de contrôle (UC) chargée de la piloter. Une structure de ce type, celle utilisée dans ce TD/TP, est illustrée en figure 4.3. La demande de division, *DIVstart*, provient d'une structure supérieure de type coeur de CPU. L'unité de contrôle (UC) locale de la division hardware génère les signaux de contrôle, pour effectuer l'algorithme de division sur l'unité de traitement (UT) dédiée. Les données sont présentées stables en entrée par le coeur de CPU, qui vient chercher les résultats de la division quand le signal *DIVend* est actif.

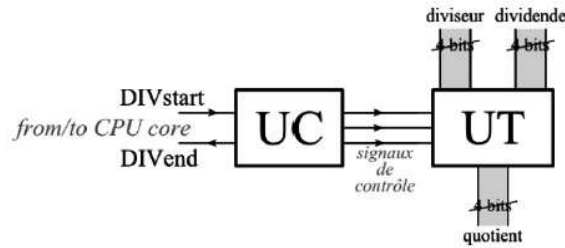


FIGURE 4.3 – Structure logique de l'algorithme de division

L'unité de traitement comprend 3 registres de données et un additionneur/soustracteur. Le registre A contient le dividende, B stocke le diviseur, P est un registre de travail.

- Le registre A possède trois commandes : **LOADA** charge le dividende, **SHIFTA** décale à gauche A d'un bit et **LOADA0** charge le bit de poids faible.
- Le registre B possède une seule commande : **LOADB** charge le diviseur.
- Le registre de travail P possède trois commandes : **RESETP** remet à zéro P, **SHIFTP** décale à gauche P d'un bit et **LOADP** charge le résultat de l'addition-soustraction.

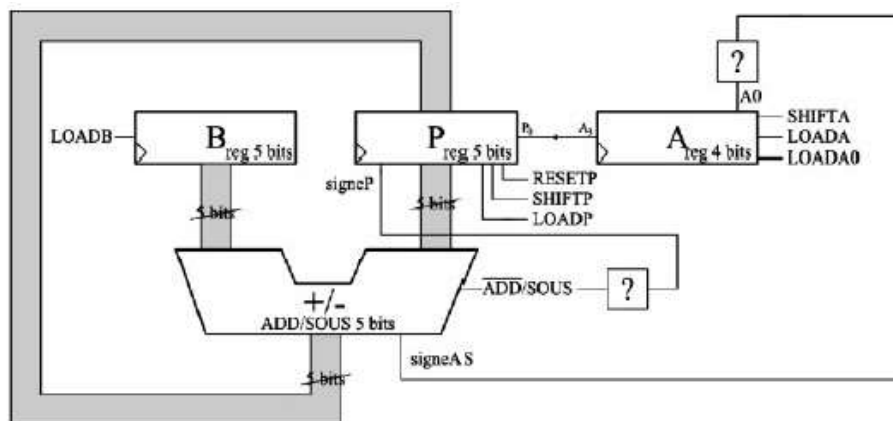


FIGURE 4.4 – Structure logique de l'unité de traitement

4.3 Réalisation de l'unité de contrôle

L'unité de contrôle doit piloter l'unité de traitement pour y effectuer l'algorithme de division. Pour l'unité de contrôle, une structure de machine à états finis de type Moore est choisie.

4.3.1 Conception de l'unité de contrôle

L'opération de division sera contrôlée par une machine d'états selon le modèle de MOORE.

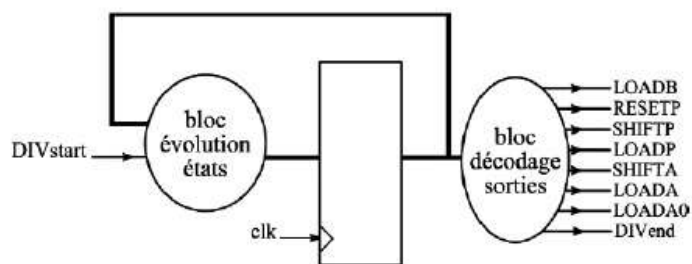


FIGURE 4.5 – Schéma de principe de l'unité de contrôle

- Le signal ***DIVstart*** démarre l'algorithme. Le registre conserve la valeur du quotient à la fin de l'algorithme.
- Une sortie ***DIVend*** indique quand le résultat est disponible.
- La période de l'horloge de cadencement, *clk*, de la machine d'état doit être supérieure au temps de calcul de l'additionneur/soustracteur.

Compétez le tableau suivant en vous inspirant de l'algorithme décrit en section 4.1.2.

Description	N°	loadB	resetP	shiftP	loadP	shiftA	loadA	loadA0	divend
début, attente divstart	0	0	0	0	0	0	0	0	0
charger A et B, reset P	1								
décalage	2								
charger A0 et P	3								
	4								
	5								
	6								
	7								
	8								
	9								
fin	10	0	0	0	0	0	0	0	1
début, attente divstart	0	0	0	0	0	0	0	0	0

Après analyse, des simplifications sont possibles et finalement, seuls 4 signaux de sortie pour l'unité de contrôle sont nécessaires pour piloter l'unité de traitement (figure 4.6).

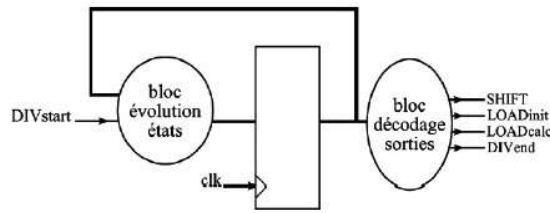


FIGURE 4.6 – Schéma de principe de l'unité de contrôle simplifiée

1. Que représentent les commandes **LOADcalc**, **LOADinit** et **SHIFT** ?
2. **Dessiner** le diagramme d'états et **coder** ses états.

4.3.2 Réalisation de la machine d'états

La réalisation proposée s'inspire directement du modèle de MOORE : 1 bloc mémoire et 2 blocs logiques combinatoires.

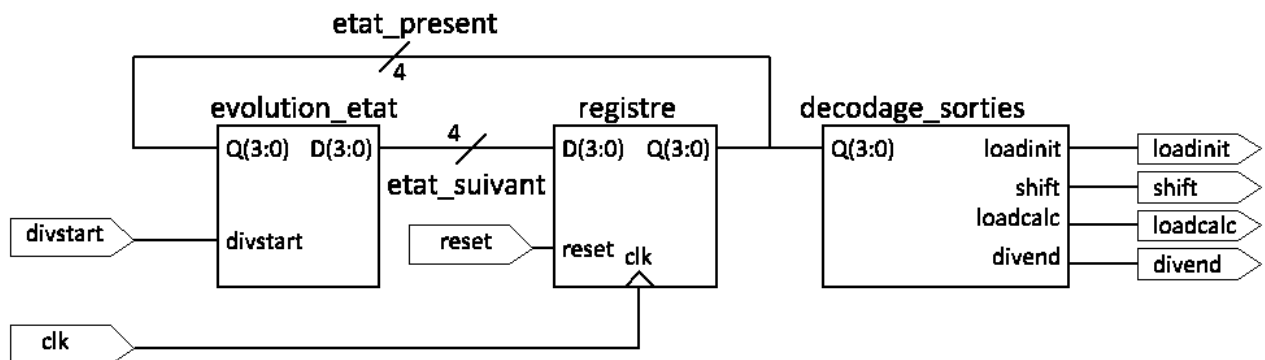


FIGURE 4.7 – Unité de contrôle

1. **Créer** un nouveau projet, nommé *division_arithmetique*.
2. **Créer** un nouveau circuit nommé *evolution_etat*. **Réaliser** le circuit *evolution_etat* à partir de sa table de vérité. **Simuler** son fonctionnement avec logisim.
3. **Créer** un nouveau circuit nommé *registre*. En utilisant les bascules disponibles dans Logisim-Evolution, **câbler** l'architecture de l'entité *registre*. **Simuler** son fonctionnement avec logisim.
4. **Créer** un nouveau circuit nommé *decodage_sorties*. **Réaliser** le circuit *decodage_sorties* à partir de sa table de vérité. **Simuler** son fonctionnement avec logisim.
5. **Créer** un nouveau circuit nommé *UC*. **Réaliser** le circuit *UC* à partir des entités *evolution_etat*, *registre* et *decodage_sorties* selon le schéma précédent.
6. **Simuler** son fonctionnement avec Questa.

4.4 L'unité de traitement

L'unité de traitement comporte un bloc combinatoire (l'additionneur) et trois registres A sur 4 bits, B sur 5bits et P sur 5bits.

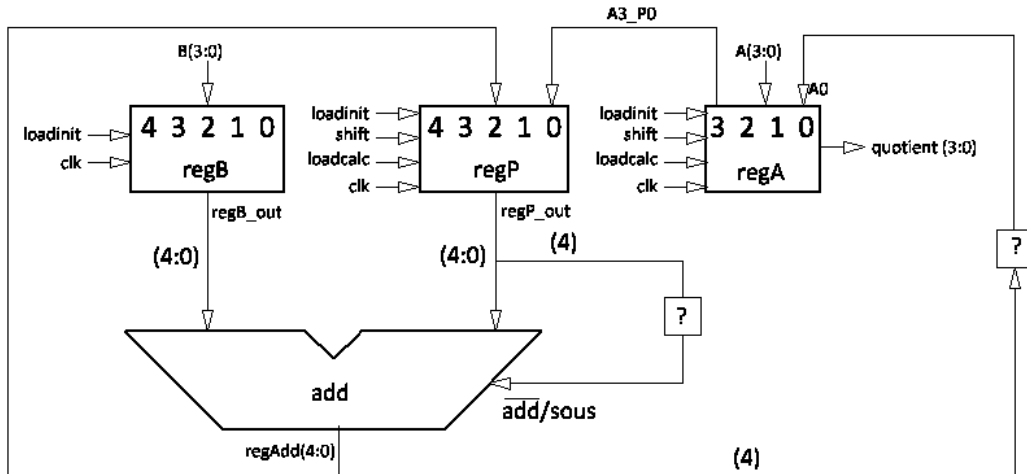


FIGURE 4.8 – Schéma de principe de l'unité de traitement

1. **Créer** un nouveau circuit nommé *regB*. Le bit de poids fort de *regB_out* sera toujours à 0.
2. **Créer** un nouveau circuit nommé *regP*.
3. **Créer** un nouveau circuit nommé *regA*.
4. **Créer** un nouveau circuit nommé *UT*.
 - (a) Quelle porte logique se trouve dans les carrés contenant un point d'interrogation ?
 - (b) **Réaliser** le circuit *UT* à partir des entités *regA*, *regB*, *regP* et *add_sous5* selon le schéma précédent.
 - (c) **Simuler** son fonctionnement avec logisim.

4.5 La division

La division, représentée par la figure suivante, consiste à assembler les 2 unités créées précédemment.

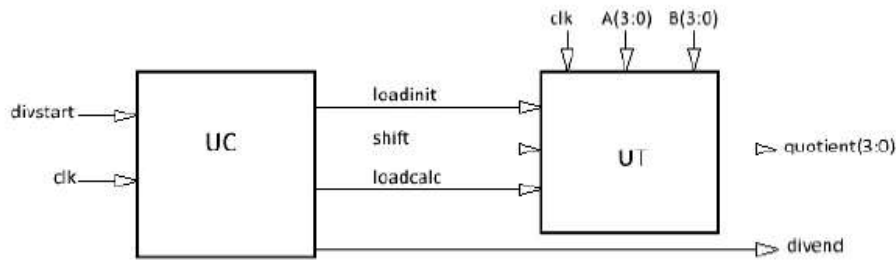


FIGURE 4.9 – Schéma de principe du montage division

1. **Créer** un nouveau circuit nommé *division_arithmetique* dont l'entité est représentée par le schéma précédent.
2. **Réaliser** le circuit *division_arithmetique* à partir des entités *UC* et *UT* selon le schéma précédent.
3. **Simuler** son fonctionnement avec logisim ou Questa.
4. **Implémenter** le composant cible et tester le fonctionnement sur la maquette.

4.6 La division avec affichage

- **Créer** un nouveau projet *division_arithmetique_afficheur*.
- **Ajouter** la librairie du TP précédent : *affichage_sequentiel_multiplexe.circ* dans laquelle se trouve le circuit *affichage_sequentiel_multiplexe*.
- **Ajouter** la librairie du TP : *division_arithmetique.circ* dans laquelle se trouve le circuit *division_arithmetique*.
- **Réaliser** le circuit *division_arithmetique_afficheur* à partir des entités *division_arithmetique*, et *affichage_sequentiel_multiplexe* qui permet d'afficher *A*, *1* et *B* sur les trois afficheurs de gauche et *quotient* sur l'afficheur de droite.
- **Implémenter** le composant cible et tester le fonctionnement sur la maquette.