



Bus internes de microprocesseur

Laboratoire Conception Numérique

Contenu

1 Objectifs	1
2 Bus de données de l'ALU	2
2.1 Circuit	2
2.2 Connexion des registres aux bus de données	2
2.3 Connexion au bus d'entrée/sortie	3
2.4 Données en provenance de l'instruction	3
2.5 Réalisation	4
3 Réalisation logicielle d'un port série	5
3.1 Transmission série	5
3.2 Algorithme	5
3.3 Réalisation	5
Glossaire	7

1 | Objectifs

Ce laboratoire exerce la conception de bus de données partagés. Il illustre l'utilisation de circuits à sortie à haute impédance.

Il présente le fonctionnement d'une unité arithmétique et logique ([Arithmetic and Logical Unit \(ALU\)](#)) de microprocesseur avec des registres de données.



2 | Bus de données de l'ALU

2.1 Circuit

La Figure Figure 1 présente une partie d'un μ processeur, composée d'une ALU, de 4 registres s_0 à s_3 , d'une interface vers un bus d'entrée/sortie (Input/Output (I/O)) et d'une connexion à l'instruction en cours d'exécution.

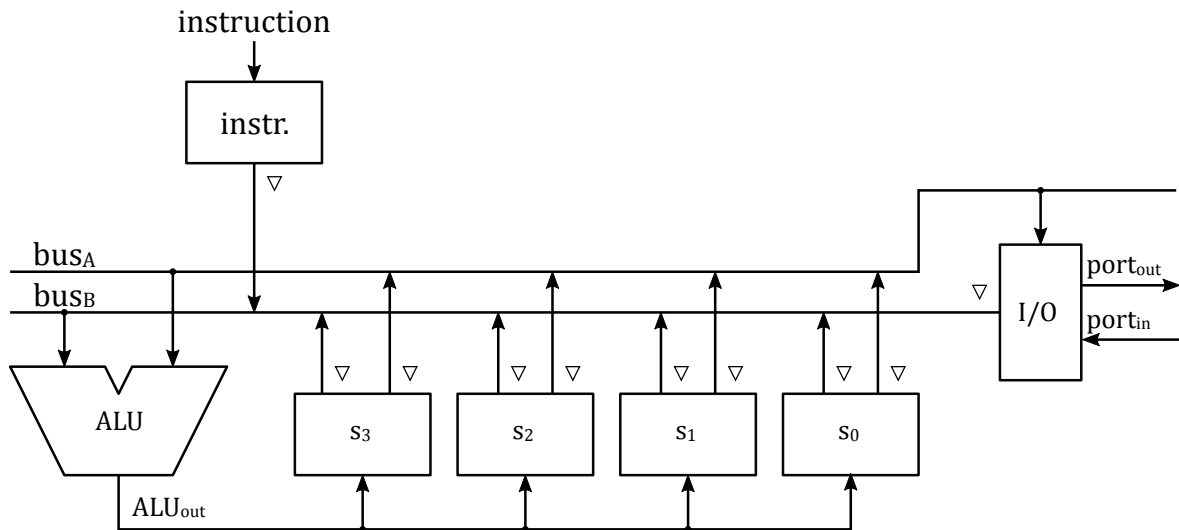


Figure 1 - ALU et registres reliés par des bus de données.

2.2 Connexion des registres aux bus de données

La Figure Figure 2 présente deux registres s_X . Au flanc montant de l'horloge, lorsque le signal de commande $en= '1'$, les données d'entrées sont chargées dans le registre: $Q^+ = D$.

Lorsque $en= '0'$, le contenu du registre est mémorisé: $Q^+ = Q$.

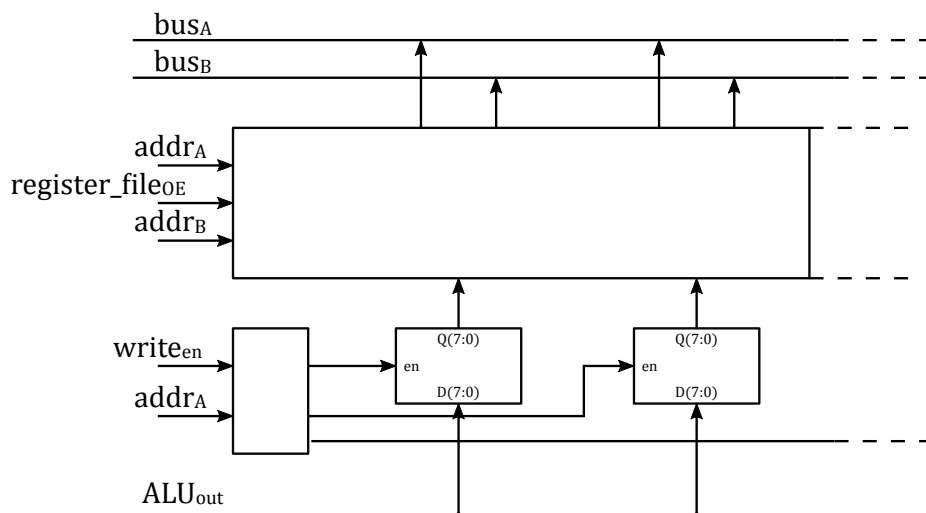


Figure 2 - Registres de données



Compléter le circuit de la Figure Figure 2 de manière à pouvoir relier la sortie des registres aux deux bus bus_A et bus_B . Le système réalisé doit simultanément permettre à un registre de placer sa donnée sur le bus_A et à un autre de le faire sur le bus_B .

Les nombres addr_A et addr_B indiquent lequel des registres transmet son information sur le bus_A , respectivement le bus_B . Le signal register_file_OE indique s'il faut amener des données du registre sélectionné sur le bus_B et permet d'éviter un conflit avec une donnée venant du port I/O ou de l'instruction.

De manière similaire, write_{en} et addr_A signalent l'écriture dans les registres. Ainsi, une opération dont le premier opérande est le registre sélectionné par addr_A verra son résultat écrit dans ce même registre.

2.3 Connexion au bus d'entrée/sortie

Dessiner le schéma interne du bloc I/O de la Figure Figure 3.

Lorsqu'une donnée est lue de l'extérieur, il faut activer le signal de commande $\text{port}_{\text{in_OE}}$ et les données du bus port_{in} sont alors écrites sur le bus_B . Lorsqu'une donnée est écrite à l'extérieur, les données du bus_A sont écrites sur le bus port_{out} et un signal extérieur à l'ALU, $\text{write}_{\text{strobe}}$, est activé au moyen du testbench. Ceci permet d'enregistrer cette donnée dans un registre externe au $\mu\text{processeur}$.

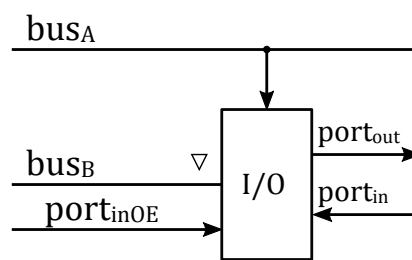


Figure 3 - Bloc d'entrée/sortie

2.4 Données en provenance de l'instruction

Pour le second opérande d'une opération, une valeur constante peut être codée dans l'instruction et amenée sur le bus_B de l'ALU. Le bloc gérant ce transfert est représenté à la Figure Figure 4.

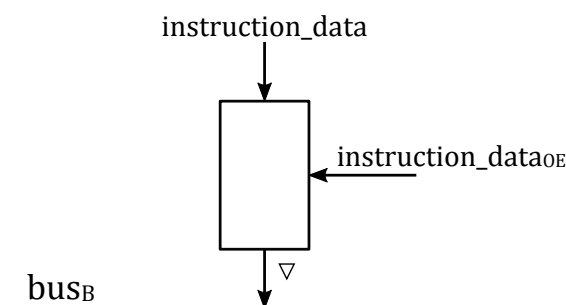


Figure 4 - Données en provenance de l'instruction

Dessiner le schéma interne du bloc de la Figure Figure 4.



2.5 Réalisation

En fonction des points précédents, compléter le circuit de bus internes de μ processeur qui vous est mis à disposition pour ce laboratoire.



3 | Réalisation logicielle d'un port série

3.1 Transmission sérielle

La Figure 5 présente le déroulement temporel de l'envoi en série d'un mot de donnée.

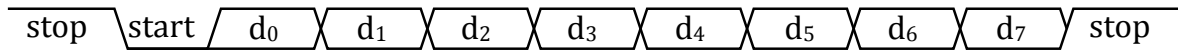


Figure 5 - Transmission sérielle

Dans notre application, le signal sériel est transmis sur le bit de poids faible du bus port_{out} . Dans le banc de test, ce bus est connecté à un registre externe. Piloté par le banc de test, le signal $\text{write}_{\text{strobe}}$ commande l'écriture dans ce registre.

3.2 Algorithme

L'algorithme à programmer est le suivant:

```

LOAD      s3, FF          ; load stop bit
OUTPUT    s3              ; output stop bit
LOAD      s3, s3          ; no operation
LOAD      s3, s3          ; no operation
LOAD      s3, s3          ; no operation
LOAD      s3, s3          ; no operation
LOAD      s0, 00          ; load start bit
OUTPUT    s0              ; output start bit
INPUT     s1              ; load word to send
OUTPUT    s1              ; output word, LSB is considered
SR0       s1              ; shift word, bit 1 -> LSB
OUTPUT    s1              ; output bit 1
SR0       s1              ; bit 2 -> LSB
OUTPUT    s1              ; output bit 2
SR0       s1              ; bit 3 -> LSB
OUTPUT    s1              ; output bit 3
SR0       s1              ; bit 4 -> LSB
OUTPUT    s1              ; output bit 4
SR0       s1              ; bit 5 -> LSB
OUTPUT    s1              ; output bit 5
SR0       s1              ; bit 6 -> LSB
OUTPUT    s1              ; output bit 6
SR0       s1              ; bit 7 -> LSB
OUTPUT    s1              ; output bit 7
LOAD      s3, s3          ; no operation
OUTPUT    s3              ; output stop bit
  
```

3.3 Réalisation

Compléter le banc de test de l'ALU pour effectuer la séquence d'instructions pour la transmission sérielle.



Il est important de ne pas laisser des bus en haute impédance. Programmer l'algorithme de manière à ce qu'il y ait toujours un signal sur les bus A et B, même lorsqu'on n'y cherche aucune information.



Glossaire

ALU – Arithmetic and Logical Unit [1](#), [2](#), [3](#), [5](#)

I/O – Input/Output [2](#)