Master SESI

# Systèmes Multi-processeurs

Cours du professeur Alain Greiner

Examen de juin 2014

**Cet examen est sans documents**

On s'intéresse dans cet exercice au contrôleur de disque respectant le protocole Pibus (pibus_block_device) utilisé dans les trois derniers TPs de l'UE MULTI (TP8,TP9,TP10).

## PARTIE A : Architecture interne du contrôleur de disque (13 points)

Du point de vue du PIbus, le contrôleur de disque est à la fois un maître et une cible. On rappelle qu'il possède 8 registres de 32 bits adressables :

- **BDEV_BUFFER** 0x00 (write-only) adresse de base du tampon mémoire.
- **BDEV_COUNT** 0x04 (write-only) nombre de blocs à transférer.
- **BDEV_LBA** 0x08 (write-only) Index du premier bloc sur le disque.
- **BDEV_OP** 0x0C (write-only) type d'opération demandée et démarrage.
- **BDEV_STATUS** 0x10 (read-only) état global du contrôleur de disque.
- **BDEV_IRQ_ENABLE** 0x14 (write-only) interruptions autorisées si non-nul.
- **BDEV_SIZE** 0x18 (read-only) retourne la capacité de stockage totale (blocs).
- **BDEV_BLOCK** 0x1C (read_only) retourne le nombre d'octets dans un bloc.

Trois types d'opérations peuvent être demandées par une écriture dans BDEV_OP :

- opération GET : lecture de BDEV_COUNT blocs de 512 octets sur le disque, et écriture des données en mémoire à l'adresse contenue dans BDEV_BUFFER.
- opération PUT : lecture de BDEV_COUNT blocs de 512 octets en mémoire à l'adresse contenue dans BDEV_BUFFER et écritures des données sur le disque.
- opération NOP : aucun transfert, et arrêt du transfert en cours.

En réponse à une commande de lecture du registre BDEV_STATUS, le contrôleur de disque peut retourner les valeurs suivantes :

- IDLE 0 : aucun transfert en cours
- BUSY 1 : transfert en cours non terminé
- GET_SUCCESS 2 : transfert type GET terminé sans erreur
- PUT_SUCCESS 3 : transfert type PUT terminé sans erreur
- GET_ERROR 4 : transfert type GET terminé, mais addresse mémoire illégale
- PUT_ERROR 5 : transfert type PUT terminé, mais addresse mémoire illégale

Toute lecture du registre BDEV_STATUS a l'effet d'un « soft reset », puisqu'elle interrompt le transfert en cours, en re-initialisant le registre BDEV_OP à la valeur NOP, et en désactivant l'interruption signalant la fin de transfert.

Outre les registres adressables ci-dessus, le composant pibus_block_device contient deux automates (TARGET_FSM et MASTER_FSM), quelques registres non visibles du logiciel, et un tampon de mémoire locale permettant de stocker un bloc de 512 octets.

L'automate **TARGET_FSM** est un automate de Moore chargé de recevoir, décoder, exécuter et répondre aux commandes de lecture ou d'écriture envoyées par le système d'exploitation vers le contrôleur de disque.

**Question A1 (1 point)** Quelle est la longueur du segment alloué au contrôleur de disque. Pourquoi l'adresse de base de ce segment doit-elle être alignée ?

**Question A2 (1 point)** Expliquez précisément pourquoi ce segment doit être non cachable.

**Question A3 (1 point)** Dans quel cas le contrôleur de disque renvoie-t-il un code d'erreur sur le PIbus en réponse à une commande de lecture ou d'écriture ?

Les signaux qui contrôlent les transitions de l'automate TARGET_FSM sont les suivants :

- SEL (1 bit) : signal PIbus indiquant que le composant est sélectionné
- READ (1 bit) : signal PIbus indiquant le sens du transfert
- A4, A3, A2 (3 bits) : bits de l'adresse PIbus désignant le registre cible

Les signaux générés par cet automate sont les suivants :

- ACK_WEN (1 bit) : signal autorisant l'écriture d'une valeur sur le bus ACK
- ACK_VAL (2 bit) : valeur réponse ACK : READY, WAIT, ERROR
- DT_WEN (1 bit) : signal autorisant l'écriture d'une valeur sur le bus DT
- DT_VAL (2bits) : valeur réponse DT : BDEV_STATUS, BDEV_SIZE, BDEV_BLOCK
- BUF_WEN (1 bit) : signal autorisant l'écriture dans le registre BDEV_BUFFER
- CNT_WEN (1 bit) : signal autorisant l'écriture dans le registre BDEV_COUNT
- LBA_WEN (1 bit) : signal autorisant l'écriture dans le registre BDEV_LBA
- OP_WEN (1 bit) : signal autorisant l'écriture dans le registre BDEV_OP
- IRQ_WEN (1 bit) : signal autorisant l'écriture dans le registre BDEV_IRQEN
- OP_RST (1 bit) : signal autorisant la remise à zéro du registre BDEV_OP

**Question A4 (1 point)** Quel composant matériel génère-t-il le signal PIBUS SEL qui indique que le contrôleur de disque est la cible d'une transaction de lecture ou d'écriture ? Comment ce composant détermine-t-il la cible de la transaction ?

L'automate TARGET_FSM décode la commande dans l'état IDLE, et renvoie la réponse dans un des autres états. Il n'introduit donc pas de cycles d'attente sur le bus.

**Question A5 (1 points)** Complétez le graphe des transitions de l'automate TARGET FSM, en attachant à chaque transaction l'expression Booléenne autorisant cette transition.

**Question A6 (1 point)** Complétez le tableau ci-dessous définissant la fonction de génération de l'automate TARGET_FSM.

L'automate **MASTER_FSM** est chargé d'effectuer les transferts de données commandés par les écritures dans les registres adressables du contrôleur de disque. Pour effectuer ces transferts, l'automate construit des rafales d'une longueur de 512 octets (1 bloc).

Outre son propre registre d'état, l'automate MASTER_FSM contrôle l'incrémentation du registre BDEV_BUFFER contenant l'adresse mémoire, et la décrémentation du registre BDEV_COUNT contenant le nombre de blocs restant à transférer. Il contrôle également un décompteur WORD_COUNT, qui décompte le nombre de mots de 32 bits dans une rafale.

Les signaux qui contrôlent les transitions de cet automate sont les suivants :

- OP (2 bits) : commande enregistrée dans BDEV_OP (3 valeurs : PUT, GET, NOP)
- GNT (1 bit) : signal PIbus indiquant que le bus a été alloué
- ACK (2 bits) : signal de réponse PIbus (3 valeurs : READY, WAIT, ERROR)
- B_LAST (1 bit) : Le décompteur de blocks a atteint la valeur 1
- W_LAST (1 bit) : Le décompteur de mots a atteint la valeur 1.

Les signaux générés par cet automate sont les suivants :

- REQ (1 bit) : signal PIbus de requête d'allocation du bus
- CMD_EN (1 bit) : autorisation d'émettre une commande sur le bus.
- READ (1 bit) : signal PIbus définissant le sens de l'échange sur le bus
- LOCK (1 bit) : signal Pibus indiquant la dernière addresse d'une rafale
- B_DEC (1 bit) : signal de décrémentation du registre BDEV_COUNT
- W_DEC (1 bit) : signal de décrémentation du registre WORD_COUNT
- W_INIT (1 bit) : signal d'initialisation du compteur WORD_COUNT
- IRQ_ACTIVE (1 bit) : signal d'interruption indiquant la fin d'un transfert

**Question A7 (1 point)** Rappelez à quoi sert le signal LOCK, en précisant quel composant matériel utilise ce signal.

L'automate MASTER_FSM demande l'allocation du bus dans les états PUT_REQ et GET_REQ. Les états PUT_AD, PUT_DTAD, PUT_DT implémentent le pipe-line pour une rafale de lectures en mémoire. De même, les états GET_AD, GET_DTAD, GET_DT implémentent le pipe-line pour une rafale d'écritures en mémoire.

L'écriture ou la lecture d'un bloc de 512 octets sur le disque est réalisé en un seul cycle dans les états PUT_BLOCK ou GET_BLOCK respectivement, ce qui impose l'existence d'une mémoire locale d'une capacité de 512 octets.

Les deux états PUT_TEST et GET_TEST permettent à l'automate de tester si le bloc qui vient d'être transféré est le dernier.

**Question A8 (1 point)** Dans quel(s) état(s) de cet automate l'interruption signalant la fin du transfert est-elle activée ? Quelle condition fait sortir l'automate de ces états ?

**Question A9 (2 points)** Complétez le graphe des transitions de l'automate MASTER_FSM en attachant à chaque transition l'expression Booléenne autorisant cette transaction. On se limitera aux transactions étiquetées A à Q.

**Question A10 (2 points)** Remplissez le tableau définissant la fonction de génération de l'automate MASTER_FSM. Attention : cet automate n'est pas d'un automate de Moore.

**Queston A11 (1 point)** Quelle est la durée d'occupation du bus pour transférer un bloc, dans l'hypothèse d'un contrôleur mémoire rapide capable de répondre sans cycles d'attente.

## PARTIE B : Pilotage du contrôleur de disque par le système d'exploitation (7 points)

**Question B1 (1 point)** Pourquoi le système d'exploitation est-il seul autorisé à programmer un transfert de données entre le disque et la mémoire.

**Question B2 (1 point)** quels sont les arguments des deux appels système ioc_read() et ioc_write() fournis par le GIET pour déclencher une opération GET ou une opération PUT On précisera le type et la signification des arguments

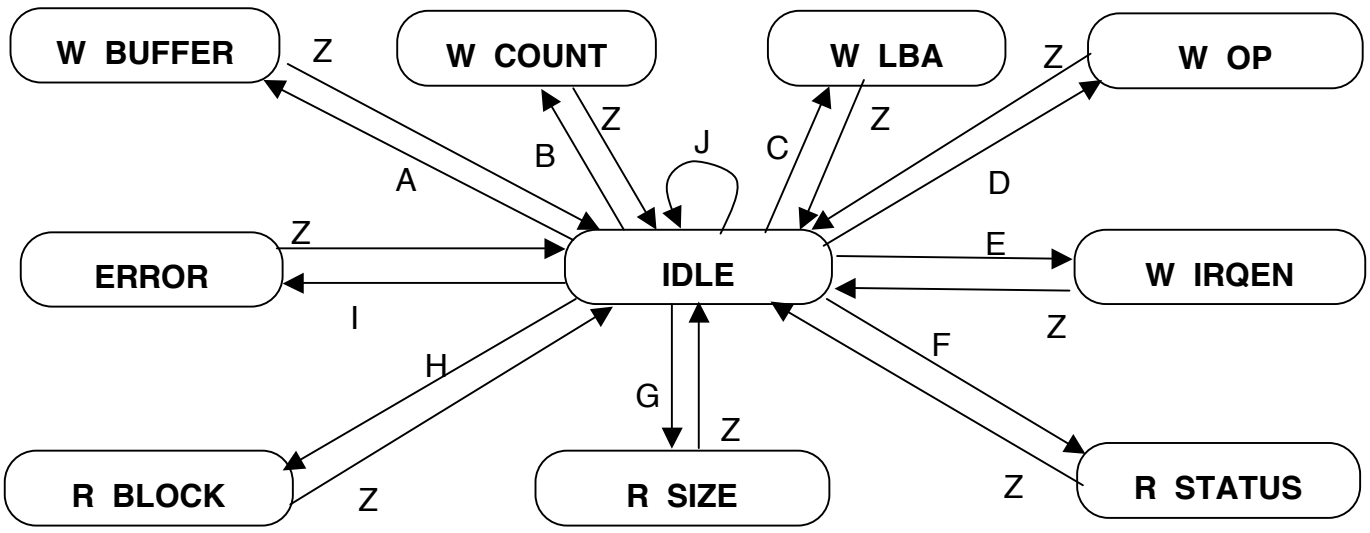
**Question B3 (1 point)** Le contrôleur de disque étudié dans la première partie est mono-canal, ce qui signifie qu'il n'est pas capable d'exécuter plusieurs transferts en parallèle. Dans une architecture multi-processeurs, comment le GIET garantit-il que deux appels systèmes exécutés par deux applications logicielles indépendantes s'exécutant sur deux processeurs différents ne déclenchent pas des accès concurrents aux registres du contrôleur de disque ? Décrivez précisément le mécanisme utilisé par le GIET pour garantir un accès exclusif.

**Question B4 (1 point)** Quelle vérification effectue le GIET avant de lancer le transfert ? Cette vérification doit être effectuée même dans le cas d'une application mono-tâche où il n'y a pas de risque d'accès concurrents.

**Question B5 (1 point)** Quels registres du contrôleur de disque doivent être accédés, et dans quel ordre, pour lancer effectivement le transfert lorsque les vérifications sont effectuées ?

**Question B6 (2 point)** Décrivez précisément le mécanisme permettant au système d'exploitation d'être informé qu'un transfert est terminé. Comment le système d'exploitation est-il informé que le transfert a échoué lorsque l'adresse du tampon mémoire appartient bien à l'espace utilisateur, mais ne correspond à aucun segment défini ? Quelle fonction libère le verrou protégeant ll'accès au contrôleur de disque ?

### Fonction de transition automate TARGET_FSM



## Fonction de génération automate TARGET_FSM

| | ACK WEN | ACK VAL | DT WEN | DT VAL | BUF WEN | CNT EN | LBA WEN | OP WEN | IRQ WEN | 0P RST |
|---|---|---|---|---|---|---|---|---|---|---|
| IDLE | | | | | | | | | | |
| W_BUFFER | | | | | | | | | | |
| W_COUNT | | | | | | | | | | |
| W_LBA | | | | | | | | | | |
| W_OP | | | | | | | | | | |
| W_IRQEN | | | | | | | | | | |
| R_STATUS | | | | | | | | | | |
| R_SIZE | | | | | | | | | | |
| R_BLOCK | | | | | | | | | | |
| ERROR | | | | | | | | | | |

## Fonction de génération automate MASTER_FSM

| | REQ | CMD_EN | READ | LOCK | W_DEC | B_DEC | W_INIT | IRQ_ACTIVE |
|---|---|---|---|---|---|---|---|---|
| IDLE | | | | | | | | |
| PUT_REQ | | | | | | | | |
| PUT_AD | | | | | | | | |
| PUT_DTAD | | | | | | | | |
| PUT_DT | | | | | | | | |
| PUT_BLOCK | | | | | | | | |
| PUT_TEST | | | | | | | | |
| PUT_SUCCESS | | | | | | | | |
| PUT_ERROR | | | | | | | | |
| GET_BLOCK | | | | | | | | |
| GET_REQ | | | | | | | | |
| GET_AD | | | | | | | | |
| GET_DTAD | | | | | | | | |
| GET_DT | | | | | | | | |
| GET_TEST | | | | | | | | |
| GET_SUCCESS | | | | | | | | |
| GET_ERROR | | | | | | | | |

**Fonction de transition automate MASTER_FSM**