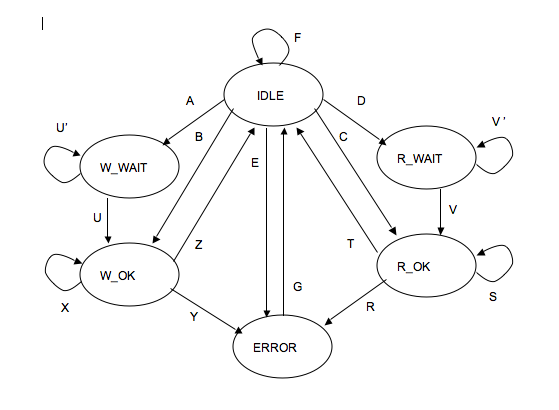
**TP1 : Prototypage virtuel & Protocole PIBUS**

**C) Automate du composant PibusSimpleRam :**

**Question C1 : Complétez le graphe représentant l'automate à 6 états qui contrôle le composant matériel PibusSimpleRam.**



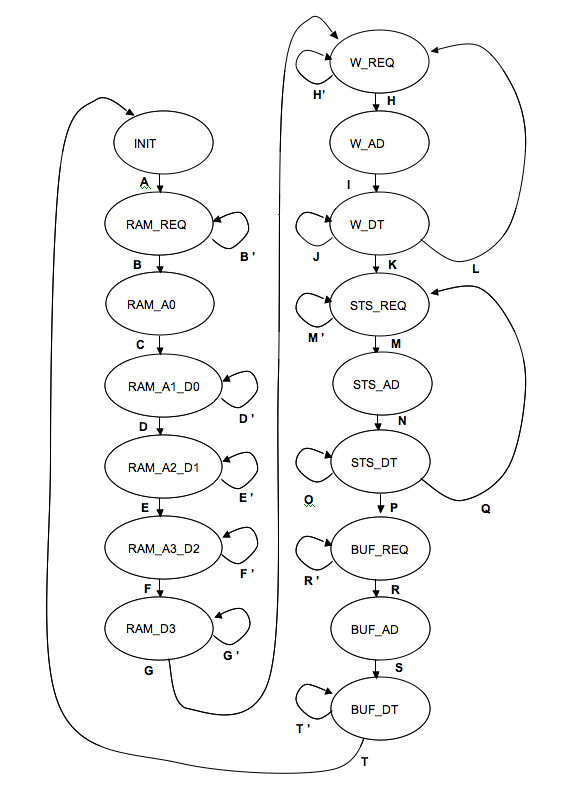
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | SELDELAYADR\_OK | X | | SEL |
| B | SELADR\_OK | Y | | SELREAD |
| C | SELADR\_OKREAD | Z | |  |
| D | SELDELAYADR\_OKREAD | R | | SEL |
| E | SEL | S | | ADR\_OKSELREAD |
| F |  | T | |  |
| G |  | U | | GO |
|  |  | | U’ |  |
|  |  | | V | GO |
|  |  | | V’ |  |

**Question C2 : Précisez la fonction de génération de cet automate (c'est à dire les signaux définissant la réponse sur le bus), en remplissant le tableau ci-dessous.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ACK\_EN | ACK\_VALUE | DT\_EN | MEM\_CMD |
| IDLE | 0 |  | 0 | NOP |
| R\_WAIT | 1 | WAIT | 0 | NOP |
| R\_OK | 1 | READY | 1 | READ |
| W\_WAIT | 1 | WAIT | 0 | NOP |
| W\_OK | 1 | READY | 0 | WRITE |
| ERROR | 1 | ERROR | 0 | NOP |

**D) Automate du composant PibusSimpleMaster :**

**Question D1 : Complêter le graphe représentant l'automate ci-dessous, en attachant à chaque transition une expression Booléenne dépendant des 4 signaux GNT, READY, LAST , et NUL**



TTY≡TELETYPE:

1) WRITE (character display) 2)STATUS (bit status keyboard) 3) READ (character keyboard)

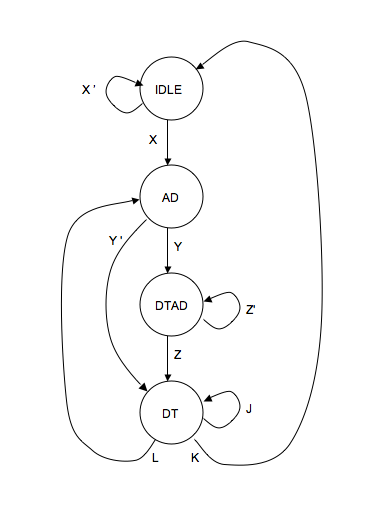
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A |  | G | RDY | N |  |
| B | GNT | G’ |  | O |  |
| B’ |  | H | GNT | P | RDY |
| C |  | H’ |  | Q | RDYNULL |
| D | RDY | I |  | R | GNT |
| D’ |  | J |  | R’ |  |
| E | RDY | K | RDYLAST | S |  |
| E’ |  | L | RDY | T | RDY |
| F | RDY | M | GNT | T’ |  |
| F’ |  | M’ |  |  |  |

**Question D2 : Précisez la fonction de génération, en remplissant le tableau définissant, pour chacun des états de l'automate, les valeurs des signaux de sortie contrôlés par cet automate.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | REQ | CMD\_EN | ADR\_VALUE | READ\_VALUE | LOCK\_VALUE | DT\_EN |
| INIT | 0 | 0 | X | X | 0 | 0 |
| RAM\_REQ | 1 | 0 | X | X | 0 | 0 |
| RAM\_A0 | 0 | 1 | RAM\_BASE | 1 | 1 | 0 |
| RAM\_A1\_D0 | 0 | 1 | RAM\_BASE+4 | 1 | 1 | 0 |
| RAM\_A2\_D1 | 0 | 1 | RAM\_BASE+8 | 1 | 1 | 0 |
| RAM\_A3\_D2 | 0 | 1 | RAM\_BASE+12 | 1 | 0 | 0 |
| RAM\_D3 | 0 | 0 | X | X | 0 | 0 |
| W\_REQ | 1 | 0 | X | X | 0 | 0 |
| W\_AD | 0 | 1 | TTY\_BASE | 0 | 0 | 0 |
| W\_DT | 0 | 0 | X | X | 0 | 1 |
| STS\_REQ | 1 | 0 | X | X | 0 | 0 |
| STS\_AD | 0 | 1 | TTY\_BASE+4 | 1 | 0 | 0 |
| STS\_DT | 0 | 0 | X | X | 0 | 0 |
| BUF\_REQ | 1 | 0 | X | X | 0 | 0 |
| BUF\_AD | 0 | 1 | TTY\_BASE+8 | 1 | 0 | 0 |
| BUF\_DT | 0 | 0 | X | X | 0 | 0 |

**E) Automate du composant PibusSegBcu :**

**Question E1 : Complétez le graphe représentant l'automate à 4 états du composant matériel PibusSegBcu ci-dessous, en attachant à chaque transition une expression Booléenne dépendant des signaux d'entrée REQ, ACK, et LOCK.**



|  |  |  |  |
| --- | --- | --- | --- |
| X | REQ | Z’ | LOCK+WAIT |
| X’ |  | L | REQ∙ |
| Y | LOCK | K |  |
| Y’ |  | J | WAIT |
| Z | ∙ |  |  |

**Question E2 : Précisez la fonction de génération, en remplissant le tableau définissant les valeurs des signaux de sortie GNT, SEL0 et SEL1 pour chacun des 4 états de l'automate.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | GNT | SEL0 | SEL1 |
| IDLE | 0 | X | X |
| AD | 1 | (A) | (A) |
| DTAD | 0 | (A) | (A) |
| DT | 0 | (A) | (A) |

1. : On choisit SEL0 ou SEL1 selon l’adresse A.

**Question E3: Expliquer pourquoi l'allocation (choix d'un maître) est réalisée non seulement dans l'état IDLE, mais aussi dans l'état DT.**

**Réponse :**Parcequ’il y a des cas où on choisit le même maître pour continuer à envoyer des données, donc il faut aussi avoir une transition de DT à AD pour réaliser l’allocation plus rapidement et plus efficacement.

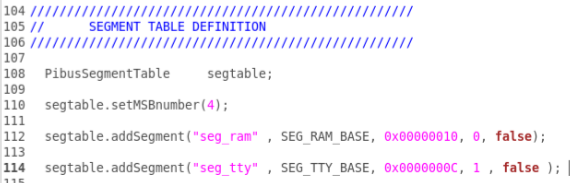
**F) Modélisation de l'architecture matérielle :**

**Question F1 : Complétez le fichier tp1\_top.cpp en instanciant et en connectant les 2 composants matériels manquant : PibusSimpleMaster et PibusSimpleRam.**

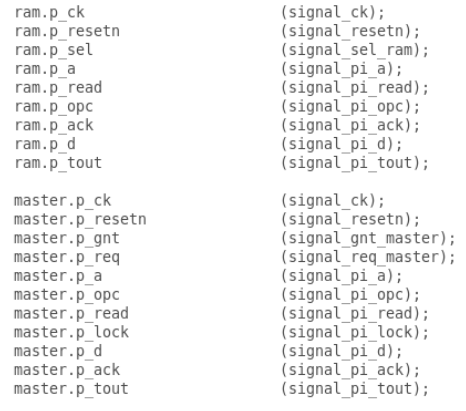
**Réponse :Des arguments dans l’instanciation sont correspondus dans les constructeurs de pibus\_simple\_ram.h et de pibus\_simple\_master.h.**



**Question F2 : Complétez le fichier tp1\_top.cpp, en définissant le segment associé au contrôleur TTY (Le segment associé à la RAM est déjà défini).**



**Réponse :**Dans l’Automate du composant PibusSimpleMaster, on peut voir que le master lit 4 mots dans RAM (16 bytes) et 3 mots dans TTY (16 bytes), donc on peut déduire les tailles des cibles ici.



**Question F3 :** **Ouvrez le fichier pibus\_simple\_ram.cpp pour déterminer comment est initialisée la chaîne de caractère "Hello World!" dans la mémoire, au début du segment seg\_ram associé à la mémoire.**

**Réponse :**On fait un schéma ci-dessous pour montrer le résultat plus clairement :

|  |  |  |  |
| --- | --- | --- | --- |
| **l** | **l** | **e** | **H** |
| **o** | **W** |  | **o** |
| **!** | **d** | **l** | **r** |
|  |  |  |  |

Mot0

début

Mot1

Mot2

Mot3

**G) Simulation**

**Question G1 : Quelle est la vitesse de simulation (mesurée en nombre de cycles simulés par seconde) ?**

**Réponse :**Après avoir lancé la simulation pour 1000000 cycles , ça fait 11.5 secondes.

Donc la vitesse de simulation=cycles/seconde.

**Question G2 : Combien y-a-t-il de cycles d'attente dans les états de l'automate du composant maître où celui-ci demande au BCU l'allocation du bus? Expliquez ce comportement.**

**Réponse :** On prends un cycle pour demander au BCU l’allocation du bus. Parce que dans cycle 1 BCU est dans l’état AD, donc on peut déduire que dans cycle 0 on a fini l’allocation du bus.

**Question G3 : Combien y-a-t-il de cycles d'attente dans les états de l'automate du composant maître ou celui-ci attend la réponse de la RAM? Expliquez ce comportement.**

**Réponse :** 2 cycles. On peut voir que RAM est dans l’état de READ\_WAIT dans cycle 2 et cycles 3.

**Question G4 : Combien faut-il de cycles pour afficher un caractère sur le composant PIBUS\_MULTI\_TTY ?**

**Réponse :** Il faut 2 cycles.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Cycle** | **Cycle** | **Cycle** |
| **BCU** | **IDLE** | **AD** | **DT** |
| **Master** | **WRITE\_REQ** | **WRITE\_AD** | **WRITE\_DT** |
| **TTY** | **IDLE** | **IDLE** | **DISPLAY** |