

# Modélisation Transactionnelle des Systèmes sur Puces

Ensimag 3A, filière SLE

Frédéric Pétrot/Julie Dumas — 7 Février 2018

## Consignes :

- Durée : 2h,
- Document autorisé : une feuille A4 manuscrite recto-verso seulement,

## 1 Questions sur C++

### 1.1 Héritage en C++

On considère le programme suivant :

```
#include <iostream>
using namespace std;

struct Base {
    const char * nv() { return "A"; }
    virtual const char * v() { return "B"; }
};

struct Derived : public Base {
    const char * nv() { return "C"; }
    virtual const char * v() { return "D"; }
};

int main() {
    Derived d;
    cout << "d.nv()_=" << d.nv() << endl;
    cout << "d.v()_=" << d.v() << endl;

    Base *pD = &d;
    cout << "pD->nv()_=" << pD->nv() << endl;
    cout << "pD->v()_=" << pD->v() << endl;

    Base b = d;
    cout << "b.nv()_=" << b.nv() << endl;
    cout << "b.v()_=" << b.v() << endl;
}
```

**Question 1** *Qu'affiche ce programme? Expliquez brièvement chaque ligne de la sortie.*

## 1.2 Méthodes virtuelles en C++

On considère le programme suivant :

```
#include <iostream>
using namespace std;

struct F {
    virtual void f() = 0;
};

struct A : public F {
    virtual void g() { cout << "appel_de_g()" << endl; }
};

struct B : public F {
    virtual void f() { cout << "appel_de_f()" << endl; }
};

int main() {
    A a;
    B b;
}
```

**Question 2** *Ce programme ne compile pas. Indiquez quelle(s) ligne(s) cause(nt) une erreur et donnez-en la cause.*

## 1.3 Macros

On considère le programme C++ ci-dessous :

```
#include <iostream>
using namespace std;

#define FOO(x) \
    do { \
        f(x+x); \
    } while (0)

#define BAR(x) \
{ \
    f(x+x); \
}

void f(int x)
{
    (volatile int)x;
}

int main() {
    int a = random()%2; // 0 ou 1
    if (a)
```

```

        BAR(a+1);
    else
        FOO(a++);

    return printf("%d\n", a);
}

```

On ne s'intéresse pas à la fonction f.

**Question 3** *Ce programme ne compile pas, expliquer pourquoi.*

**Question 4** *Si l'on inverse les lignes FOO et BAR, il compile, expliquer pourquoi. Il reste néanmoins un problème dont gcc vous informera par un warning. Identifiez-le.*

## 2 Modélisation du temps et ordonnancement en SystemC

On considère le programme suivant :

```

#include <systemc>
#include <iostream>

using namespace std;
using namespace sc_core;

SC_MODULE(A) {
    sc_event e;
    void f() {
        cout << "f1" << endl;
        e.notify();
        wait(e);
        cout << "f2" << endl;
        sleep(3);
        e.notify();
        wait(1, SC_SEC);
        cout << "f3" << endl;
        e.notify();
        wait(e);
    }
    void g() {
        cout << "g1" << endl;
        e.notify();
        wait(e);
        cout << "g2" << endl;
        e.notify();
        sleep(1);
        wait(e);
        wait(3, SC_SEC);
        cout << "g3" << endl;
        wait(e);
        e.notify();
    }
    SC_CTOR(A) {

```

```

        SC_THREAD(f);
        SC_THREAD(g);
    }
};

int sc_main(int, char**) {
    A a("a");
    sc_start();
    return 0;
}

```

On rappelle que la fonction `sleep(N)` est une fonction POSIX (pas une fonction SystemC) qui provoque une attente de `N` secondes du processus courant.

La norme SystemC autorise plusieurs exécutions possibles de ce programme (i.e. l'ordonnanceur a la liberté de choisir entre ces exécutions).

**Question 5** *Combien y a-t-il d'exécution différentes (i.e., produisant des affichages différents) autorisées par SystemC? Donnez ces traces d'exécution.*

**Question 6** *Représentez toutes les exécutions sur un graphique à deux dimensions. On mettra le "wall-clock time" sur l'axe des ordonnées et le temps simulé sur les abscisses. Le schéma fera apparaître  $f_1, f_2, f_3, g_1, \dots, g_4$ . Si on néglige le temps de calcul, combien de temps prendra chaque exécution en "wall-clock time"? Combien de temps en temps simulé?*

### 3 Implantation d'un accélérateur matériel

Dans cette partie, nous allons ajouter un accélérateur matériel à la plate-forme du TP2 (affichage d'une image en niveau de gris). L'objectif (pas très crédible) de cet accélérateur est de recopier l'image en place (not comment sur l'utilité) en en calculant si demandé le « négatif ». Pour ce faire, l'accélérateur, qui est un maître/esclave, va lire chaque pixel de l'image un par un, en prendre le complément, et de le réécrire à sa place.

En pseudo code, ça donne ceci, en supposant que `negate` est un booléen :

```

for (int y = 0; y < HEIGHT * WIDTH / 4; y++) {
    read(a, rvalue);
    wvalue = negate ? ~rvalue : rvalue;
    write(a, wvalue);
    a += 4;
}

```

Nous allons maintenant modéliser un composant matériel qui implante ce (délicat) algorithme. Le composant se nomme `Negatif`, possède comme adresse de base sur le bus l'adresse `NEGA_BASEADDR` (macro dont on dispose n'importe où) et contient un unique registre, `negate`, à l'*offset* 0, qui est accessible en écriture seulement! Écrire un 0 (zéro) dans ce registre fera une copie directe, alors qu'un 1 (un) provoquera la copie avec calcul du négatif. C'est l'action d'écrire qui provoque le lancement de la copie, i.e., pas d'écriture dans ce registre, pas d'action du composant.

Le modèle TLM du composant `Negatif` est donné par la classe ci-dessous :

```

struct Negatif : sc_core::sc_module {
    ensitlm::initiator_socket<Negatif> initiator_socket;
    ensitlm::target_socket<Negatif> target_socket;
    sc_core::sc_event go_go_go;
    SC_CTOR(Negatif);
    void compute();
    tlm::tlm_response_status write(ensitlm::addr_t a, ensitlm::data_t d) {
        switch (a) {
            case ZERO :
                ...
                return tlm::TLM_OK_RESPONSE;
            default:
                SC_REPORT_ERROR(name(), "write_register_not_implemented");
                return tlm::TLM_ADDRESS_ERROR_RESPONSE;
        }
    }

    tlm::tlm_response_status read(ensitlm::addr_t a, ensitlm::data_t& d) {
        ...
    }
    bool negate;
};

```

**Question 7** *Expliquez les rôles respectifs des sockets initiateur et cible du composant Negatif.*

**Question 8** *À quoi sert selon vous l'événement go\_go\_go? Écrivez le code correspondant au case ZERO du switch.*

Nous allons maintenant implanter la méthode `compute` du composant. On en donne une ébauche ci-dessous, dans laquelle `VRAM_BASE` est l'adresse de base de la mémoire vidéo :

```

void Negatif::compute() {
    while (true) {
        ensitlm::addr_t a = VRAM_BASE;
        ensitlm::data_t rvalue, wvalue;
        /* TODO: ... */

        for (int y = 0; y < HEIGHT * WIDTH / 4; y++) {
            /* TODO: negate or not negate ! */
            ...
            a += 4;
        }

        cout << name() << ":_finished_image" << endl;
    }
}

```

**Question 9** *Le calcul effectué étant celui indiqué au début de l'exercice, proposez une encapsulation SystemC qui respecte les spécifications énoncées sur le lancement d'une copie et l'éventuel calcul du négatif lors de celle ci.*

**Question 10** Ajoutez à la fonction `sc_main` le nécessaire à l'instanciation de ce composant.

On donne ci-dessous le corps de la fonction `generator` telle que définie dans le TP2. La fonction `test`, donnée, produit l'image en RAM vidéo, tout comme dans le TP.

```
void Generator::compute() {
    socket.write(LCDC_BASE + LCDC_ADDR_REG, VRAM_BASE);
    socket.write(LCDC_BASE + LCDC_START_REG, 0x000000001);

    while (true) {
        if (!interrupt)
            wait(interrupt_event);
        interrupt = false;

        socket.write(LCDC_BASE + LCDC_INT_REG, 0x00000000);
        test(0); // Transfère l'image en RAM Vidéo
    }
}
```

**Question 11** Que faut-il ajouter dans ce code pour qu'une copie ait lieu à chaque nouveau transfert de l'image, et que 64 images affichées soient « directes » puis 64 « négatives », et cela continuellement? Astuce : jouez avec les bits!