# Introduction à SystemC

## Historique et concepts

Tarik Graba

P4 2015-2016


Télécom ParisTech

# Table des matières

# 1 Présentation de l'UE

## Objectifs

- Présenter un nouveau *langage* de description du matériel et de modélisation ***SystemC***
- Présenter des concepts de modélisation à différents niveaux d'abstraction
- Regarder sous le capot (SystemC est opensource)

## Organisation et évaluation

- Cours suivi de mise en pratique
- Travaux à faire avant la séance suivante
- QCM à la fin (la date n'est pas encore fixée)

# 2 Introduction

## Généralités

- SystemC est un “langage” de modélisation et de description du matériel.
  - SystemC est en réalité une bibliothèque `C++`
- SystemC est un standard ouvert (Norme IEEE-1666)
- Une implémentation de référence, libre (open source) est distribuée par Accellera.
- *SystemC permet de décrire du matériel en* `C++`

---

- Le standard est distribué sous la forme d’un manuel de référence (Language Reference Manual) et est disponible gratuitement sur le site de l’IEEE (Institute of Electrical and Electronics Engineers). Il peut être téléchargé sur cette page du site de l’IEEE.
- Accellera System Initiative est une organisation à but non lucratif dont l’objectif est de développer et de promouvoir des standards industriels pour la conception et la modélisation de systèmes électroniques. La liste des standards soutenus par cette organisation se trouve ici.
- L’implémentation de référence de SystemC est accessible librement (après acceptation de la licence d’utilisation) sur la page dédié du site d’Accellera. Pour ce cours nous utiliseront la version 2.3 du standard (la version 2.3.1 propose des fonctionnalités en avance par rapport à la norme qui sont désactivées par défaut).
- Sur les machines des l’école, la bibliothèque est accessible dans le répertoire `/comelec/softs/opt/systemc-2.3.0`. Le répertoire contient le source ainsi qu’une version compilée pour Linux x86_64.

---

## Historique

- 1998, Synopsys “ouvre” son outil Scenic et crée SystemC 0.9
- 2000, Standard OSCI (Open SystemC Initiative)
  - Avec des contributions de Frontier Design et CoWare
- Standard IEEE en 2005 avec la version 2.0
  - SystemC V2.2 correspond au standard IEEE 1666-2005

- Fusion OSCI/Accellera en 2011 et mise à jour du standard
  - SystemC V2.3 correspond au standard IEEE 1666-2011

*SystemC plus de 15 ans d'évolution !*

---

À la base, c'est une contribution de plusieurs entreprises qui est devenu un standard.

La première version de SystemC apportait déjà tous les éléments nécessaires pour la description du matériel :

- Un simulateur évènementiel
- Des processus
- Des signaux

La version 2 de SystemC a apporté des mécanismes de modélisation plus haut niveau dits transactionnels. D'abord sous la forme d'une extension (TLM : *Transaction Level Modeling*) qui a été ensuite intégrée au standard. La version 2.3 de SystemC apporte la $2^{e}$ version de cette extension (TLM 2.0).

CoWare et Frontier Design, n'existent plus. Elles ont disparu dans les multiples fusions et rachats entre les différents acteurs du marché de l'EDA.

---

# Pourquoi SystemC

- Pourquoi utiliser `C++` pour décrire du matériel ?
- Pour modéliser *efficacement* un système complet contenant du logiciel et du matériel.

---

Dans un système sur puce (SoC : *System On Chip*) il y a du matériel :

- des processeurs,
- des accélérateurs (graphiques, audio…),
- des interfaces de communication

et du logiciel exécuté sur ce matériel.

Il faut donc un moyen efficace de modéliser l'ensemble.

Cette modélisation a pour but de :

- Concevoir le système :
  - développer les algorithmes,
  - définir ce qui est fait en logiciel et ce qui est fait en matériel (partitionnement),
- Simuler le système :
  - vérifier les fonctionnalités,

  - avoir un modèle de référence.

C++ a été choisi car :

- c'est un langage objet et on peut s'appuyer sur cela pour représenter des *modules* matériels,
- c'est un langage *efficace* avec lequel on peut obtenir de très bonnes performances,
- c'est un langage que beaucoup de développeurs connaissent déjà.

---

# À quel niveau d'abstraction ?

- Dit autrement, qu'avons-nous besoin de modéliser pour décrire un système numérique complet contenant du logiciel et du matériel ?

# Que veut dire *niveau de représentation* ?

- Dans un modèle, avec quel niveau de précision doit-on représenter le comportement ?
- Le temps est-il important ?
  - Oui : On parle de *Timed Model*
  - Non : On parle de *Untimed Model*
- Tous les signaux sont importants ?
  - Oui : On parle de modèle *Bit Accurate*
  - Non : On parle de modèle *Transactionnel*
- A-t-on besoin d'être précis *dans* le modèle ou seulement aux interfaces ?

---

**Pour la fonctionnalité :**

Pour décrire un algorithme, on n'a besoin de décrire que la succession d'opérations menant au résultat. Dans cette description *fonctionnelle* le temps nécessaire à l'exécution de ces différentes étapes n'a pas besoin d'apparaître. Ce type de modèle est en général un point de départ pour définir les différentes fonctionnalités que notre système devra réaliser. On parle de modèle *Untimed Functionnal* (UTF).

Dès qu'on veut évaluer les performances d'une implémentation pour connaître, par exemple :

- la puissance de calcul nécessaire (fréquence, nombre d'opérateurs…),
- le nombre d'opérations qu'on peut effectuer dans un temps donné,

on doit introduire une notion de temps.

On doit donc avoir au moins, une description fonctionnelle avec une information de temps sur chaque étape. On parle de modèle *Timed Functionnal* (TF).

**Pour l'interface :**

Pour décrire des échanges de données dans le système, il peut suffire de les décrire en termes de transactions. Des lectures ou des écritures entre différents éléments du système (un processeur, une mémoire…). S'il y a plusieurs transactions en parallèle, il faut pouvoir garantir l'ordre dans lequel elles ont lieu.

Si en plus, on veut pouvoir vérifier que ces transactions respectent un certain protocole de bus, il faut que le modèle décrive précisément tous les signaux de contrôle.

---

## Que veut dire *niveau de représentation* ?

---

Dans la conception d'un système sur puce (*SoC*) on passe par plusieurs étapes. À partir d'un modèle fonctionnel, on sépare l'application en partie logicielle et partie matérielle. On parle de *partitionnement*.

Ensuite, les modèles de chaque partie sont *raffinés*.

Pour la partie logicielle, on va utiliser des modèles du support d'exécution de plus en plus précis en :

- utilisant un modèle du système d'exploitation (*OS*),
- faisant apparaître une notion de temps d'exécution,
- en utilisant des plateformes virtuelles ou des simulateurs de jeu d'instruction (ISS : *Instruction Set Simulator*).

Pour la partie matérielle, on va également passer par plusieurs étapes. À partir d'un modèle architectural, dans lequel on définit les différents blocs utilisés, on passe à des modèles transactionnels faisant apparaître les différents transferts de données puis les temps nécessaires pour les traitements et les transferts.

Ces modèles deviennent de plus en plus précis au cours du développement jusqu'à obtenir un modèle précis au cycle et au bit près. Ce modèle est dit CABA (*Cycle Accurate Bit Accurate*) qui peut servir de référence à l'écriture d'un modèle RTL synthétisable.

Tout au long de ce *flot de développement* on doit aussi conserver un *modèle de référence* pour vérifier la conformité avec l'application de départ.

**Remarque :**

Le modèle RTL est un modèle CABA dans lequel, en plus d'être précis au cycle près et au bit près, on doit respecter un style de codage particulier.

---

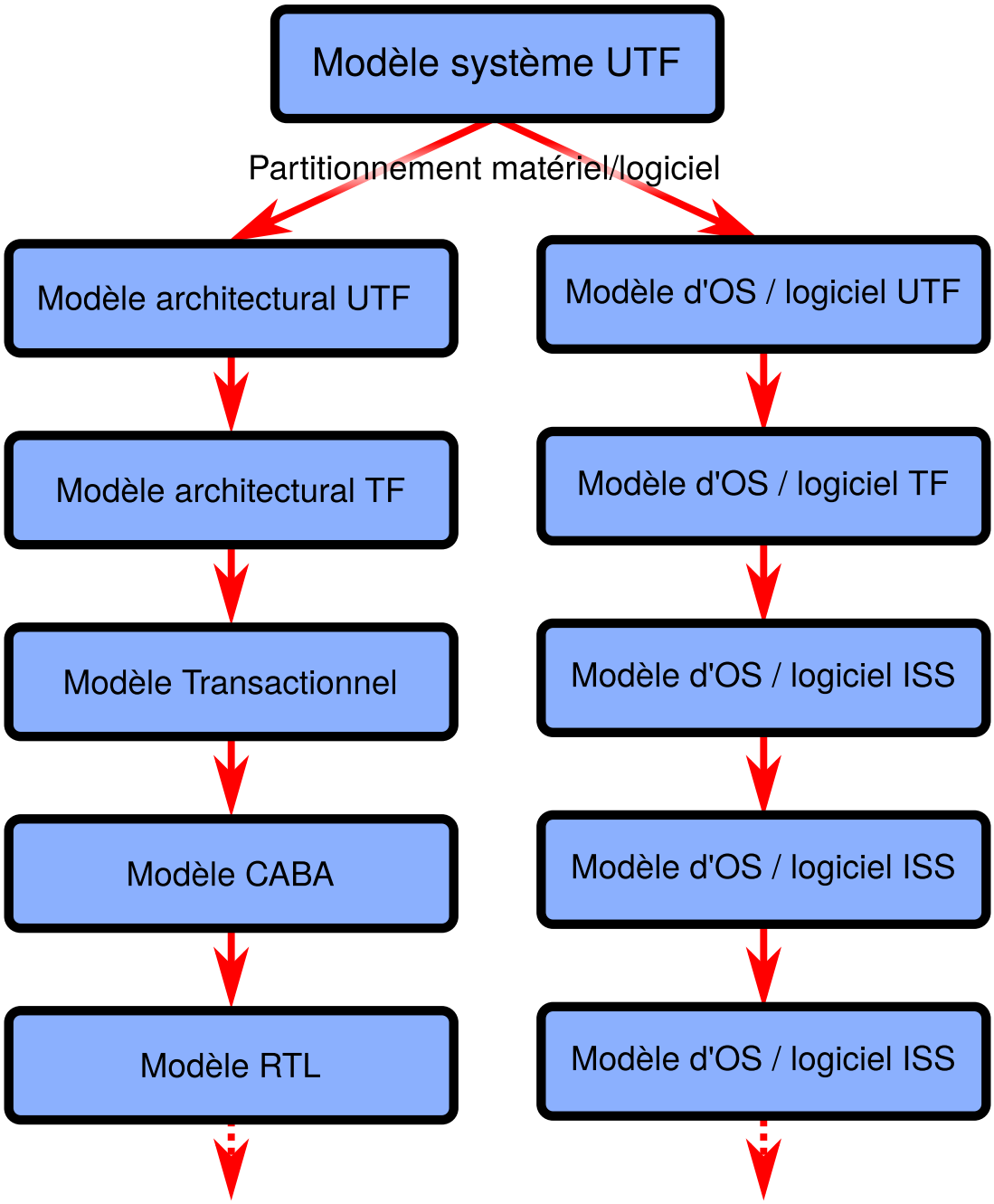


FIG. 2.1: Niveaux d'abstraction dans un flot de conception

## SystemC : Un langage pour tous les modèles

L'idée derrière SystemC est d'avoir un langage unique pour écrire des modèles à ces différents niveaux d'abstraction.

**Avantages :**

- Un seul langage à connaître.
- Faciliter le passage d'un niveau à l'autre et même son automatisation.
- Simplifier en réduisant le nombre de modèles.
- Éviter d'introduire des erreurs en retranscrivant manuellement.

## SystemC : Un langage pour tous les modèles

**Pourquoi est-ce possible en SystemC ?**

- C'est du C++ donc du logiciel.
- On peut utiliser *facilement* des bibliothèques logicielles du système.
- On peut concevoir des modèles qui vont jusqu'au niveau RTL.
- On peut mélanger des modèles de niveaux différents.

## SystemC : Un langage pour *presque* tous les modèles

Dans la *réalité* SystemC n'est pas utilisé pour tout faire. Pourquoi ?

---

Dans l'industrie, SystemC est principalement utilisé comme langage de modélisation. Il est rarement utilisé pour produire des modèles fonctionnel haut niveau ni pour produire des modèles au niveau RTL.

Actuellement, SystemC est utilisé pour la conception de modèles transactionnels permettant de simuler efficacement des systèmes sur puce. Par exemple des modèles de :

- processeurs,
- contrôleurs mémoire,
- interconnexion…

L'utilisation de SystemC permet de concevoir des modèles dont on peut faire évoluer la précision jusqu'à des modèles CABA. Ces modèles peuvent alors être interfacés avec des modules écrits dans d'autres langages de description tels que SystemVerilog ou VHDL.

---

## SystemC : Un langage pour *presque* tous les modèles

**Des habitudes**

- Il existe des langages spécialisés dans certains domaines (Matlab…) qui facilitent la vie des développeurs.
- Tout le monde ne veut pas apprendre le C++.
- Il y du code qui existe déjà !

**Les outils**

- Les outils de synthèse RTL pour SystemC n'ont jamais été développés (ça peut changer)
- Le passage d'un niveau à l'autre n'est pas vraiment automatique.

# 3 SystemC

## La bibliothèque

SystemC est une bibliothèque logicielle écrite en C++ qui contient les éléments nécessaires à la modélisation d'un système matériel numérique.

C'est-à-dire, ce qu'il faut pour simuler le parallélisme du matériel :

- un moteur de simulation événementiel, des évènements,
- des *signaux*, des types *logiques*,

Plus des extensions pour :

- la modélisation transactionnelle,
- la vérification…

---

Il faut noter qu'il existe depuis 2010 une extension AMS (*Analog/Mixed-signal*) de SystemC pour la modélisation des systèmes analogiques et mixtes. Cette extension utilise un moteur de simulation analogique différent du moteur événementiel et des concepts propres à la simulation analogique qui ne seront pas abordés ici.

---

## Structure de la bibliothèque

---

SystemC est une bibliothèque construite au-dessus de C++ donc tout ce qui est disponible en C++ l'est aussi en SystemC. Ceci est important pour pouvoir réutiliser des bibliothèques logicielles existantes qu'elles aient été écrites en C ou en C++.

Au-dessus de C++, un moteur de simulation événementiel est implémenté. Ce *moteur* permet de gérer les processus et les évènements comme dans d'autres langages HDL (Verilog ou VHDL).

L'implémentation open source de référence distribuée par Accellera contient aussi le moteur de simulation implémenté en C++.

SystemC définit un certain nombre de types utiles à la modélisation du matériel, parmi lesquels :

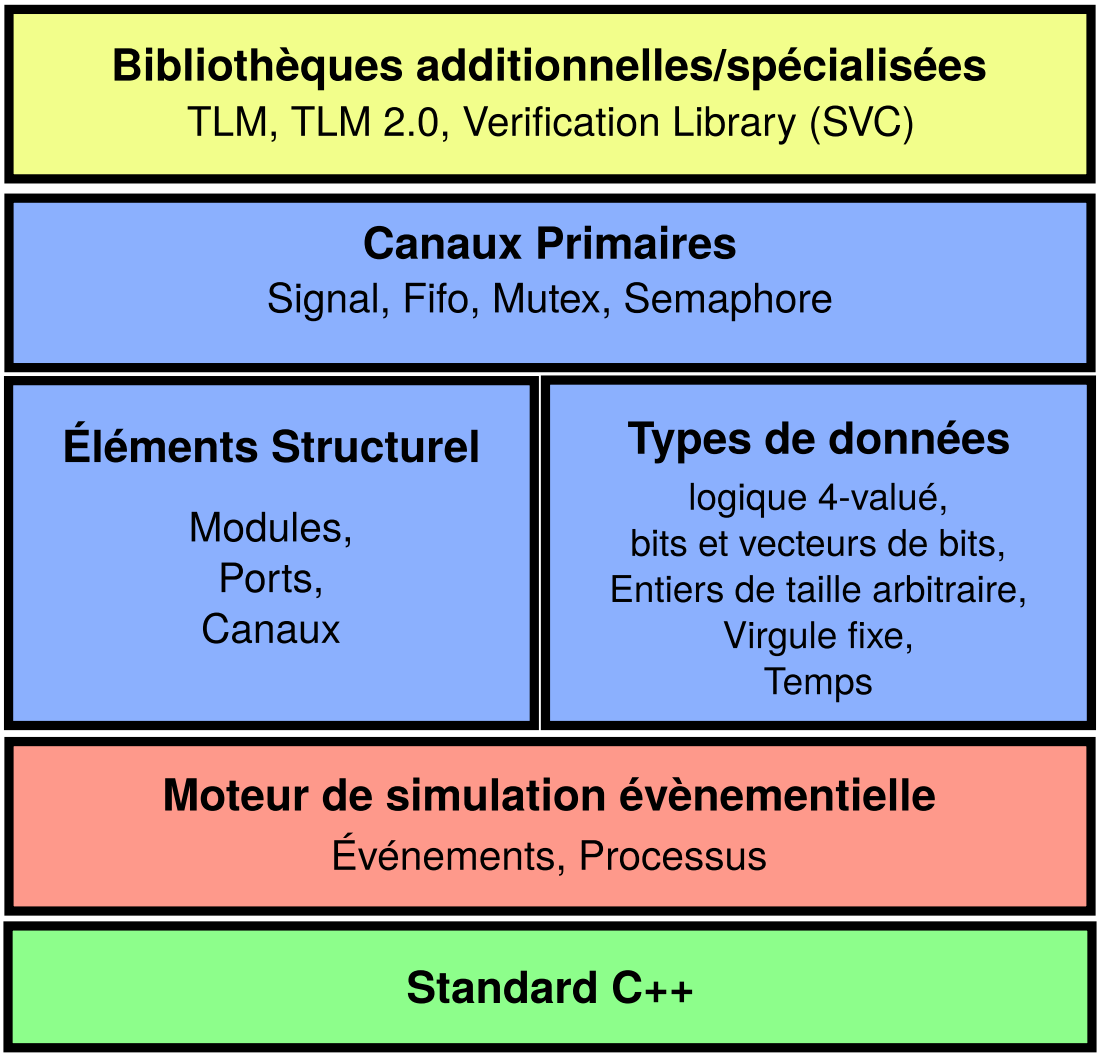


FIG. 3.1: Organisation de la bibliothèque SystemC

- des types multi-valué (0,1,x,z),
- des bits et des vecteurs,
- des entiers de taille arbitraire,
- et même des nombres en représentation en virgule fixe.

Sont aussi définis les éléments nécessaires à la description hiérarchique : modules, ports et canaux.

En SystemC, les canaux de communication vont au-delà des simples signaux pour permettre des niveaux d'abstraction plus élevés. Un certain nombre de canaux primaires existe dans la bibliothèque (signaux, Fifo…) mais des canaux supplémentaires peuvent être définis si besoin.

Au-dessus de tout ça des bibliothèques supplémentaires sont implémentées. Certaines sont fournies avec l'implémentation de référence telles que les bibliothèques pour la modélisation transactionnelle (TLM/TLM2.0) d'autres sont indépendantes comme la bibliothèque pour la vérification (SVC).

D'autres bibliothèques basées sur SystemC peuvent être fournies par des tiers (vendeur d'outils, d'IP…).

---

## Comment récupérer SystemC

La bibliothèque est disponible sur le site d'Accellera :

http ://accellera.org/downloads/standards/systemc

Peut être installée sur Linux, Mac OSX et Windows et peut être compilée au moins avec g++, clang++ ou Visual C++.

Est déjà installée sur les machines de l'école.

---

La procédure d'installation est décrite en détail dans le fichier INSTALL dans l'archive de la bibliothèque.

Extrait de ce fichier :

```
System Requirements
===================

SystemC can be installed on the following UNIX, or UNIX-like platforms :

    o 32-bit Linux (x86) (RedHat Enterprise Linux 4, 5, 6; Fedora 14, 15;
      Debian 5.0; Ubuntu 10.04LTS, 12.04LTS) with GNU C++ compiler versions
      gcc-3.3.2 through gcc-4.7.0
    o 32-bit Linux (x86) (Debian, Ubuntu 12.04LTS) with Clang C++ compiler
      version clang-2.9 through clang-3.1
    o 64-bit Linux (x86_64) (RedHat Enterprise Linux 4, 5, 6; Fedora 17;
      Debian 5.0; Ubuntu 10.04LTS, 12.04LTS) with GNU C++ compiler versions
```

```
    gcc-3.4.5 through gcc-4.7.0
  o 64-bit Linux (x86_64) (Ubuntu 12.04LTS) with Clang C++ compiler
    version clang-3.1
  o 64-bit Linux (x86_64) (Debian 5.0) with 32-bit GNU C++ compiler version
    gcc-4.1.1 through gcc-4.7.0 (./configure --target=i686-linux-gnu)
  o Sun Solaris 9 (sparc32) with GNU C++ compiler version gcc-3.3.3
    through gcc-4.4.6
  o 32-bit Mac OS X (ppc, i386) (10.5 Leopard, 10.6 Snow Leopard, 10.7 Lion)
    with GNU C++ compiler versions gcc-3.1 through gcc-4.2.1
  o 64-bit Mac OS X (x86_64) (10.6 Snow Leopard, 10.7 Lion) with
    GNU C++ compiler version gcc-4.2.1
  o 32-bit FreeBSD 7.4 with GNU C++ compiler version gcc-4.2.1
  o Windows 7 SP1, Microsoft Visual C++ 2005 SP1 (8.0)
  o Windows 7 SP1, Microsoft Visual C++ 2008 SP1 (9.0)
  o Windows 7 SP1, Microsoft Visual C++ 2010     (10.0)
  o Windows XP Prof. SP3, with Microsoft Visual C++ 2010 (10.0)
  o Windows XP Prof. SP3 (Cygwin 1.5.25) with GNU C++ compiler
    version gcc-4.3.2
  o Windows XP Prof. SP3 (Msys 1.0.16) with MinGW32
    GNU C++ compiler version gcc-4.5.2 through gcc-4.6.2

An installation of SystemC on other, related platforms may be
possible as well.
```

Sauf si vous avez un compilateur C++ extraterrestre, vous devriez pouvoir utiliser SystemC.

---

## Premier exemple *Hello World*

**hello.cpp**

```
#include <systemc.h>

int sc_main (int argc, char * argv[])
{

   cout << "hello world" << endl;

   return 0;
}
```

La fonction principale n'est pas `main` mais `sc_main`.

---

La fonction `main` existe quand même, elle est fournie par la bibliothèque SystemC. Elle s'occupe de configurer l'environnement de simulation avant d'appeler la fonction `sc_main` en lui passant les arguments qu'elle a reçus.

Le fichier d'en-tête `systemc.h` contient les définitions des éléments de la bibliothèques. En plus,

- il inclue d'autres fichiers d'en-tête de la bibliothèque standard C++, par exemple `iostream`
- et permet d'utiliser directement certains espaces de noms (`namespace`) relatifs à SystemC et à la bibliothèque standard (`std`).

Si on a besoin de gérer précisément les en-têtes et les espace de noms on peut préférer inclure `systemc`.

---

## Que doit-on compiler ?

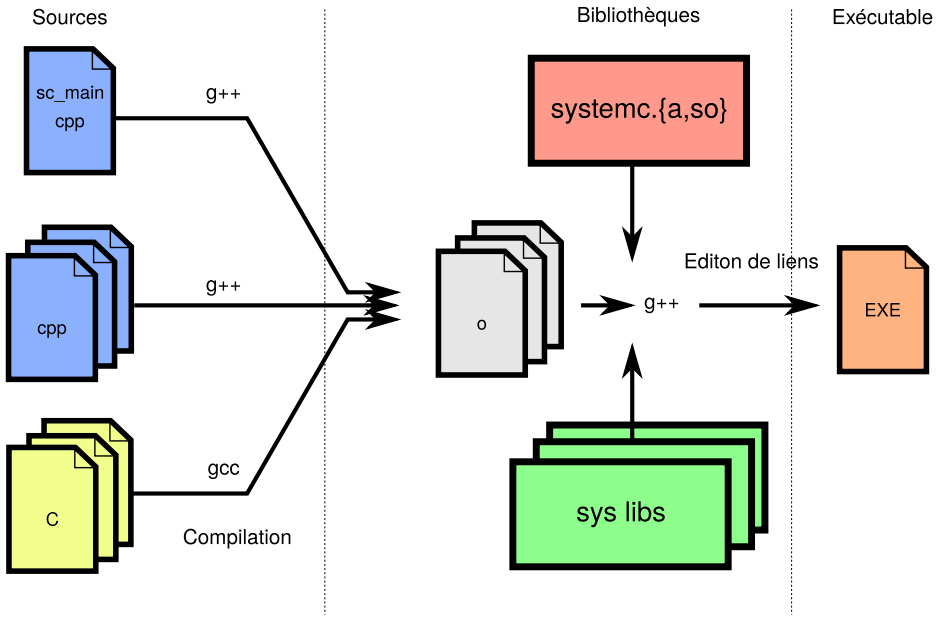


FIG. 3.2: Flot de compilation

---

Les sources C++ et C sont compilés puis l'édition de liens est faite pour générer un exécutable. Parmi ces sources, doit figurer la définition de la fonction `sc_main`.

La bibliothèque SystemC doit être fournie pour l'édition de liens. Elle apporte les éléments suivants :

- la fonction `main`,
- le simulateur événementiel,

- les autres objets de la bibliothèque SystemC.

Comme, il s'agit d'un flot de compilation standard pour du C++, on peut faire appel à d'autres bibliothèques du système.

---

## Première compilation

**Makefile**

```
SYSTEMC  ?= /comelec/softs/opt/systemc-2.3.0
ARCH      = linux64

CPPFLAGS  =  -isystem $(SYSTEMC)/include
CXXFLAGS  =  -Wall -g
LDFLAGS   =  -L$(SYSTEMC)/lib-$(ARCH)
LDLIBS    =  -lsystemc
```

---

Dans ce `Makefile` minimaliste pour `g++` :

- On précise les chemins de recherche pour les fichiers d'en-tête et les bibliothèques précompilées.
- On ajoute la bibliothèque `systemc` au moment de l'édition de liens.

L'exécution donne :

```
    SystemC 2.3.0-ASI --- Mar 20 2014 16:38:27
Copyright (c) 1996-2012 by all Contributors,
ALL RIGHTS RESERVED

hello world
```

Le message sur la version de SystemC est issu de la bibliothèque et montre bien que des choses sont faites avant l'appel à `sc_main`.

---