1. La Structure de Données Lofig

    1. Fonctions membres de la class Locon
    2. Fonctions membres de la class Losig
    3. Fonctions membres de la classe Loins
    4. Fonctions membres de la classe Lofig

La Structure de Données Lofig

La structure de données {{Lofig}} est conçue pour représenter les *netlists* en mémoire. Les attributs des différents objets constituants cette structure ont été présentés en cours. Nous allons maintenant compléter avec les déclarations des fonctions membres.

Fonctions membres de la class Locon

Les constructeurs :

- Un `Locon` peut appartenir, soit à un modèle (`Lofig`), soit à une instance (`Loins`). Il y aura donc deux constructeurs correspondants à chacune de ces possibilités. On fournit en outre son nom et sa direction. Le type sera déduit à partir du constructeur appelé.
    - `Locon ( Lofig*, const std::string& name, unsigned int dir );`
    - `Locon ( Loins*, const std::string& name, unsigned int dir );`

Les accesseurs :

- `std::string getName ();`
- `Losig* getSignal ();`
- `Lofig* getModel ();`
- `Loins* getInstance ();`
- `unsigned int getDirection();`

Les modificateurs (*mutators*, en VO) :

- Un `Locon` peut (doit) être associé à un signal. On pourra le faire de deux façons différentes, soit en donnant explicitement un pointeur sur le `Losig`, soit en indiquant simplement le nom du signal.
    - `void setSignal ( Losig* );`
    - `void setSignal ( const std::string& );`
- Positionnement de la direction :
    - `void setDirection ( unsigned int );`

De plus, le code de trois fonctions membres vous sont fournies pour pouvoir afficher l'objet dans un flux. Dans `Locon.h` :

```
class Locon {
  public:
    static std::string   typeToString ( unsigned int );
    static std::string   dirToString  ( unsigned int );
  public:
            void          xmlDrive     ( std::ostream& );
};
```

Code de Locon.cpp

Fonctions membres de la class Losig

Le constructeur :

- `Losig ( Lofig*, const std::string&, unsigned int type );`

Les arguments correspondent respectivement à la `Lofig` auxquel le signal appartient, son nom et son type. L'identificateur sera demandé directement à la `Lofig` dans le corps du constructeur.

Les accesseurs :

- `Lofig* getOwner ();`
- `std::string& getName ();`
- `unsigned int getId ();`
- `unsigned int getType ();`

Le code de la fonction utilitaire d'affichage dans un flux : Losig-Skeleton.cpp

Fonctions membres de la classe Loins

Le constructeur :

- `Loins ( Lofig* owner, Lofig* model, const std::string& );`

Les arguments sont l'`owner`, la `Lofig` *dans laquelle* l'instance est crée, le `model`, la `Logig` *dont* on créé une instance et le nom de cette instance. Le constructeur se charge de la duplication des connecteurs du modèle `model` dans l'instance.

Les accesseurs :

- `std::string& getName ();`
- `Lofig* getModel ();`
- `Lofig* getOwner ();`
- `std::list<Locon*>& getConnectors ();`
- `Locon* getConnector ( const std::string& );`

Les modifieurs :

- `bool connect ( const std::string& name, Losig* );`

Réalise l'association entre un connecteur (`Locon`) *de l'instance* et un signal `name` (de la `Lofig` `owner`).

Le code de la fonction utilitaire d'affichage dans un flux : Loins-Skeleton.cpp

Fonctions membres de la classe Lofig

Le constructeur :

- `Lofig ( const std::string& );`

Le constructeur est extrèmement simple, il se contente de positionner le nom de la `Lofig`, tous les autres attributs sont *vides*.

Le destructeur :

- `~Lofig ()`

Voir les considération spécifiques sur la destruction de la base de données.

Les accesseurs :

- `unsigned int getMode ();`
- `const std::string& getName ();`
- `std::list<Lofig*>& getModels ();`
- `std::list<Locon*>& getConnectors ();`
- `std::list<Loins*>& getInstances ();`
- `std::list<Losig*>& getSignals ();`

Les modificateurs :

- `setMode ( unsigned int );`

Les modificateurs relatifs aux modèles :

- Ajoute un nouveau modèle, soit en donnant directement sa `Lofig`, soit par nom. On n'ajoutera pas un modèle s'il est déjà présent.
    - `void addModel ( const std::string& );`
    - `void addModel ( Lofig* );`
- Retrait d'un modèle de la liste. Mêmes variantes que pour l'ajout.
    - `void removeModel ( const std::string& );`
    - `void removeModel ( Lofig* );`
- Recherche d'un modèle par son nom. S'il n'est pas trouvé, la fonction renverra `NULL`.
    - `Lofig* findModel ( const std::string& );`

Les modificateurs relatifs aux connecteurs :

- Ajoute un nouveau connecteur à la `Losig`. On doit fournir son nom, sa direction ainsi que le signal auquel il est relié. Cela implique que l'on créé les signaux *avant* les connecteurs.
    - {{{void addConnector (const std::string&, unsigned int dir, Losig*);
- Suppression d'un connecteur, soit directement par un pointeur sur le `Locon`, soit par nom.
    - {{{void removeConnector (const std::string&);
    - {{{void removeConnector (Locon*);
- Recherche d'un connecteur par son nom.
    - {{{Locon* findConnector (const std::string&);
- Réalisation d'une connexion. La première version correspond à la (re)connexion d'un signal sur un `Locon` du modèle. La seconde version à la connexion d'un `Locon` d'une instance (référencé par son nom `conName`) à un signal du modèle référencé par son nom `sigName`.
    - {{{bool connect (const std::string& conName, const std::string& sigName);
    - {{{bool connect (const std::string& insName, const std::string& conName, const std::string& sigName);