

topologyGUI

Guide d'utilisateur



Cédric Gormond - *Etudiant à Télécom Saint Etienne*

Le code source disponible sur : https://github.com/cedric-gormond/topologyGUI_win
Contact : **cedric.gormond@gmail.com**

Sommaire

Sommaire.....	2
1 Informations à propos de topologyGUI	2
2 Introduction.....	3
3 Rappel : Blocs, Topologies et Surfaces.....	4
3.1 Bloc.....	4
3.2 2D maillée.....	4
3.3 2D hexagonale	4
3.4 3D maillée.....	5
3.5 Surface d'une topologie 2D maillée.....	6
3.6 Surface d'une topologie 2D hexagonale	6
3.7 Surface d'une topologie 3D maillée.....	7
4 Comment importer un fichier de contrainte ?	8
5 Les fichiers de contrainte simplifiés	9
5.1 Le fichier simplifié original	9
5.2 Le fichier simplifié	9
6 Comment modifier d'un fichier de contrainte ?	10
7 Comment créer un fichier de contrainte ?	11
7.1 Création de topologies 2D et 3D.....	11
7.2 Remarques à propos de la topologie 3D	11
8 Comment gérer les surfaces d'une topologie ?	12
8.1 Comment calculer la surface d'une topologie ?	12
8.2 Comment calculer la distance à partir d'une surface ?	12
9 Comment récupérer un fichier de contrainte généré ?	13
10 Logs	13
11 Bugs	13

1 Informations à propos de topologyGUI

topologyGUI a été développé en C++ sous l'IDE Clion (Mac) et Visual Studio (PC) à partir de :

- **SFML** : Framework open-source permettant de rendus graphiques (*licence 'as-is'*).
 - o <https://www.sfml-dev.org/community.php>
- **ImGui** : Librairie graphique open-source permettant l'utilisation d'un GUI (*licence MIT*)
 - o <https://github.com/ocornut/imgui>
- **ImGui-SFML** : Librairie permettant l'intégration de la bibliothèque ImGui au Framework SFML (*licence MIT*)
 - o <https://github.com/eliasdaler/imgui-sfml>

2 Introduction

TopologyGUI permet de créer et/ou modifier des fichiers de contraintes d'architecture « Network On Chip » (NoC) pour le logiciel Xilinx Vivado. Ces fichiers de contraintes contiennent l'emplacement de routeurs (blocs) qui ont été placé aléatoirement ou approximativement par l'utilisateur sur Vivado. Cela permet ainsi d'établir des topologies homogènes très rapidement.

TopologyGUI a pour but :

- Utiliser un fichier de contrainte existant, généré auparavant par Vivado, pour l'importer et modifier ses caractéristiques, notamment la disposition des routeurs.
- Créer de nouveaux fichiers de contrainte en précisant la disposition des routeurs et leurs caractéristiques.
- Gérer les surfaces des topologies.

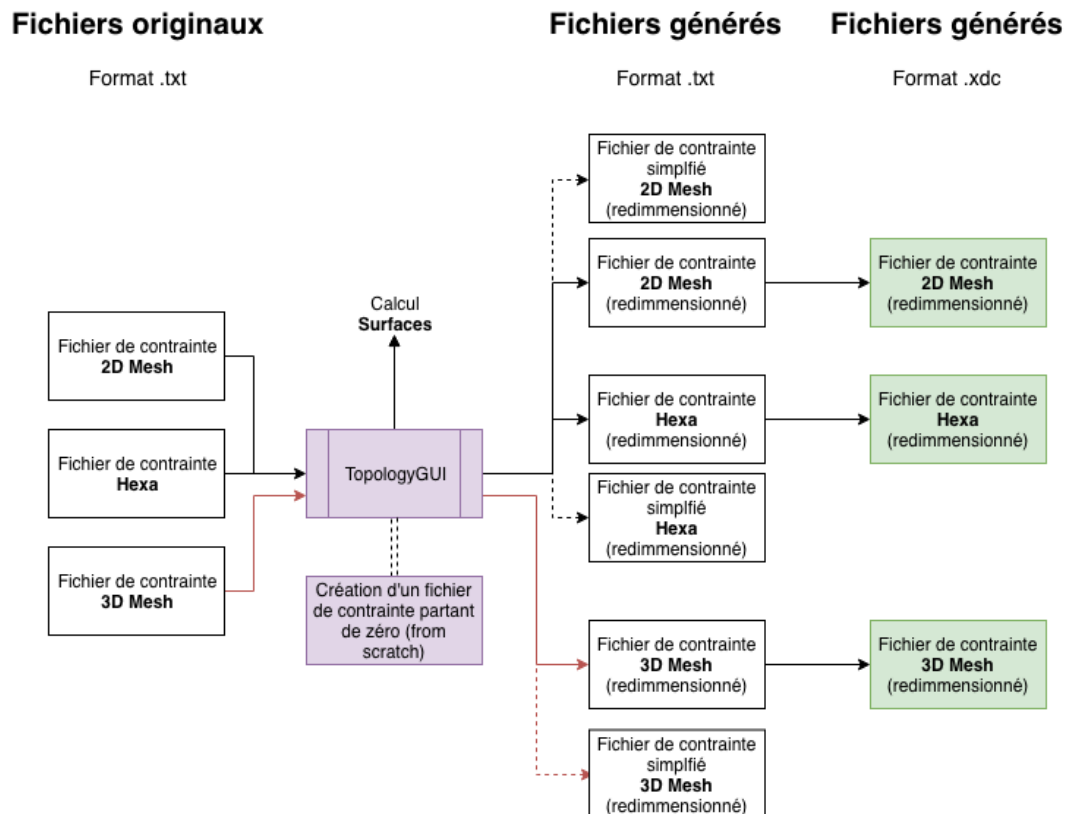


Figure 1 : Schéma fonctionnel

3 Rappel : Blocs, Topologies et Surfaces

Cette partie est à titre indicative et permet de définir différentes notions utilisées dans le logiciel pour une meilleure compréhension et prise en main. Pour passer cette partie, allez à la p8

3.1 Bloc

Un bloc est un routeur de l'architecture des NoC. Sous le logiciel Vivado, ceux-ci sont appelés *pblocs*. Un bloc est un rectangle défini à partir de la position de son coin inférieur gauche (**X0,Y0**) et la position de son coin supérieur droit (**X1,Y1**). Un bloc possède une hauteur **h** et une largeur **w**.

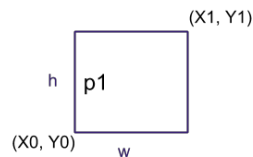


Figure 2 : Bloc

3.2 2D maillée

L'utilisateur peut établir ou créer une topologie 2D maillée en spécifiant la distance (slices) **entre le centre de chaque bloc**. La hauteur et la largeur de chaque bloc est basée sur celle du premier bloc (**pbloc1**). Il est possible de conserver la hauteur et largeur du bloc1 par défaut ou de les redimensionner, comme illustré sur Figure 3 avec w' et h' .

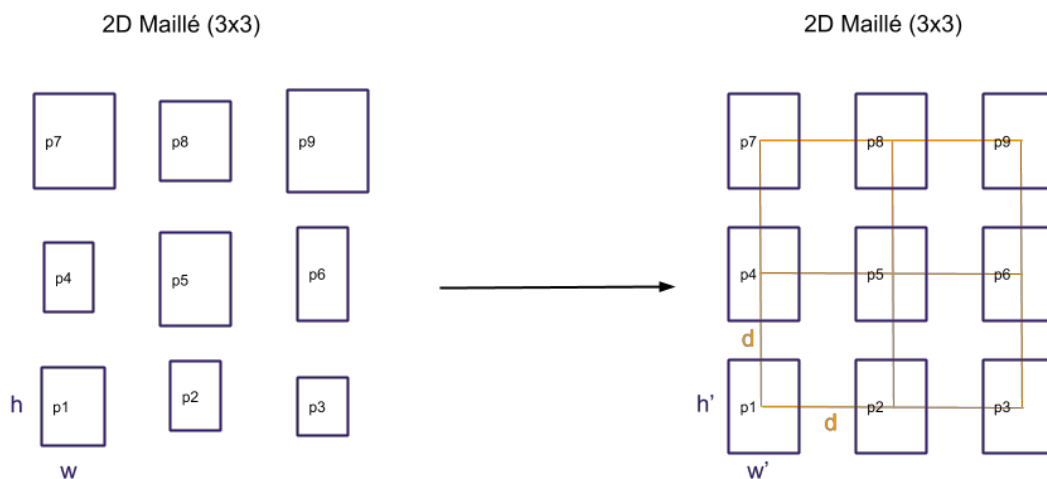


Figure 3 : 2D maillée

3.3 2D hexagonale

L'utilisateur peut établir ou créer une topologie 2D hexagonale en spécifiant la distance r (slices) entre **le centre du bloc1 et les centres des blocs autour**. La hauteur h' et la largeur w' de chaque bloc est basée sur celle du premier bloc. Il est donc possible de conserver ces données ou de les redimensionner. La distance d est $r \cos(30)$ (voir [Figure 4](#)).

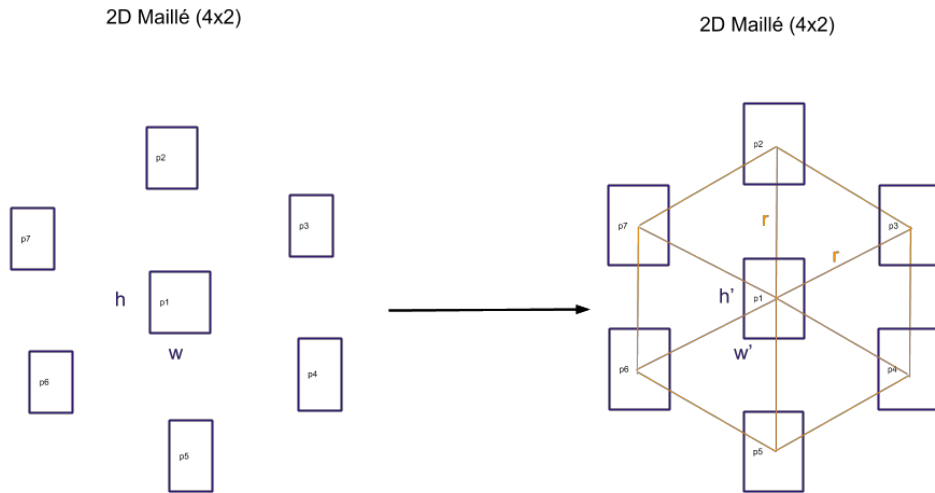


Figure 4 : Topologie hexagonale

Dans la plupart des cas, certains routeurs ne seront pas utilisés pour la topologie hexagonale. En effet, nous remarquons sur la Figure 5 que les routeurs P1, P7, P13 et P6, P12 sont en superflus dans la topologie. Il est recommandé d'utiliser au minimum 9 blocs.

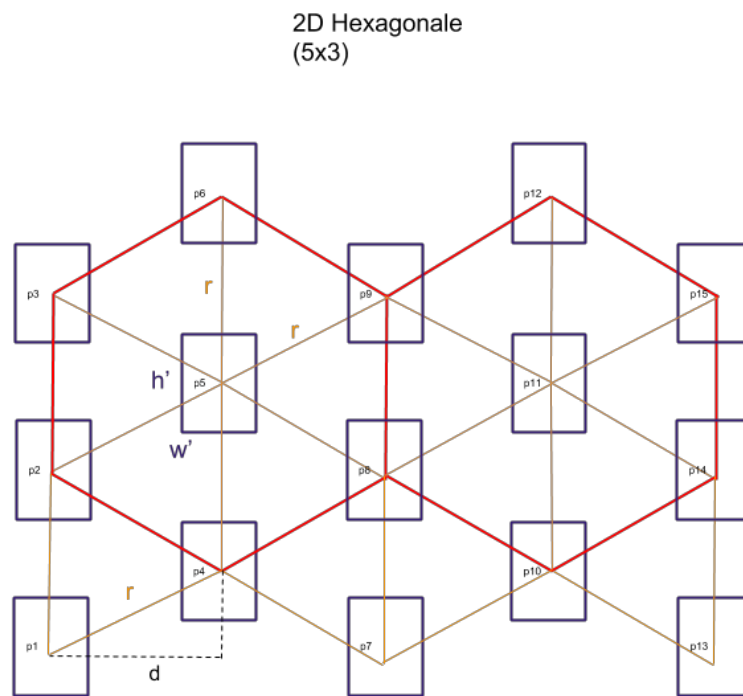


Figure 5 : Topologie hexagonale 5x3

3.4 3D maillée

L'utilisateur peut établir ou créer une topologie 2D hexagonale en spécifiant la distance (slices) entre le centre de chaque bloc. Cependant, l'utilisateur est limité à **seulement une profondeur** ($Z < 2$). **La hauteur et la largeur de chaque bloc est basée sur celle du premier (pbloc1).** Il est donc possible de conserver ces données ou de les redimensionner. Les blocs en profondeur commencent avec une coordonnées $x_0 + \frac{d}{2}$ et $y_0 + \frac{d}{2}$ (voir Figure 6).

3D Maillé (3x3x2)

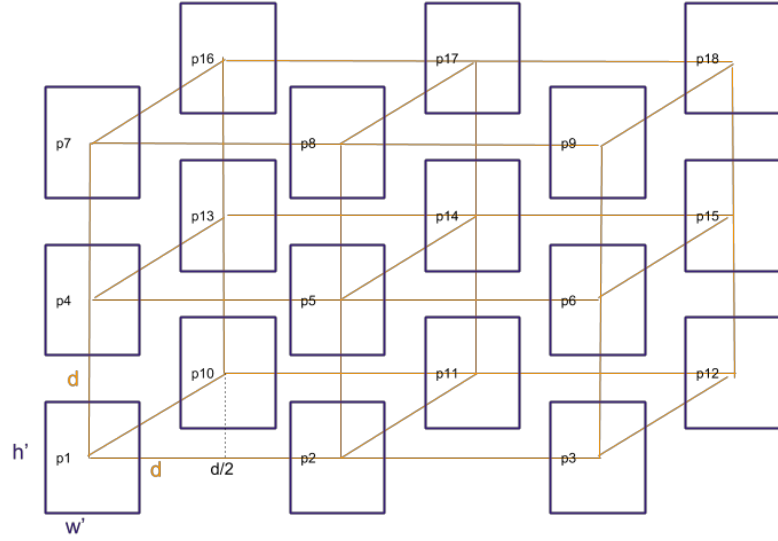


Figure 6 : Topologie 3D maillée

ATTENTION ! Il n'est pas possible de générer une topologie 3D maillée à partir d'une topologie 2D maillée ou hexagonale, et inversement.

3.5 Surface d'une topologie 2D maillée

Pour une topologie 2D maillée, la surface se calcule en faisant le produit de la largeur totale $W = (n - 1)d + w'$ par la hauteur totale $H = (m - 1)d + h'$ avec $n = \dim X$ et $m = \dim Y$.

2D Maillé (3x3)

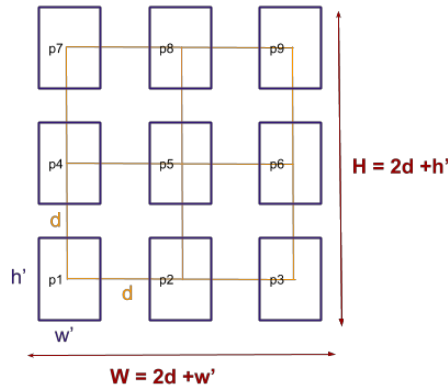


Figure 7 : surface maillée 2D

3.6 Surface d'une topologie 2D hexagonale

Pour une topologie 2D maillée, la surface se calcule en faisant le produit de $W = (n - 1)d + w' = (n - 1)(r \cos(30)) + w'$ par la hauteur totale $H = (m - 1)r + h'$ avec $n = \dim X$ et $m = \dim Y$.

2D Hexagonale
(3x3)

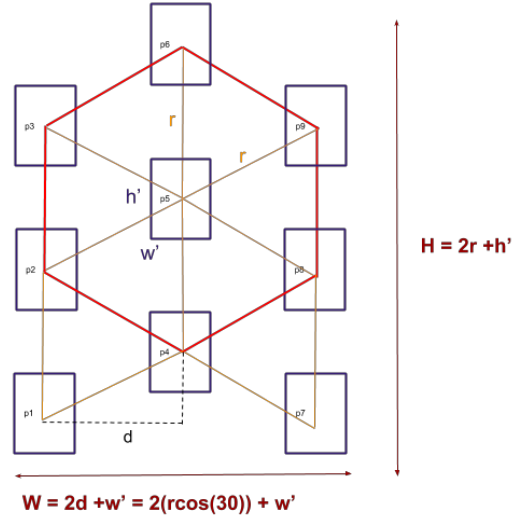


Figure 8 : Surface 2D hexagonale 3x3

3.7 Surface d'une topologie 3D maillée

Pour une topologie 3D maillée, la surface se calcule par le produit de $W = d \left((n - 1) + \frac{1}{2} \right) + w'$ par la hauteur totale $H = d \left((m - 1) + \frac{1}{2} \right) + h'$ avec $n = \dim X$ et $m = \dim Y$.

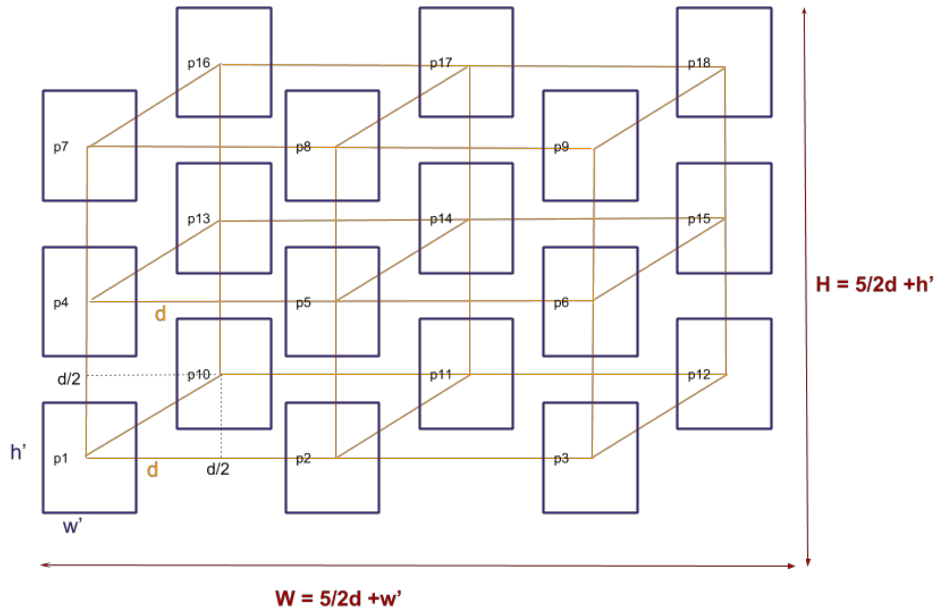


Figure 9 : surface 2D maillée 3x3x2

4 Installation

Vous devez extraire le contenu de l'archive dans un espace de travail. Pour utiliser le programme, vous devez lancer l'exécutable contenue dans le dossier.

5 Comment importer un fichier de contrainte ?

Il faut premièrement modifier l'extension du fichier de contrainte au format **.txt** (le logiciel n'arrive pas encore à reconnaître le format. xdc). Ensuite, pour importer un fichier, il faut que celui-ci soit présent dans le répertoire **io** dans répertoire contenant l'exécutable. **Attention, le fichier de contrainte doit avoir le nom suivant :**

cstr_file.txt

Exemple :

▼	folder	topologyGUI	aujourd'hui à 09:27
▶	folder	cmake_modules	3 décembre 2018 à 16:23
▶	folder	cmake-build-debug	8 février 2019 à 09:51
▶	folder	cmake-build-release	8 février 2019 à 09:51
▶	folder	imgui	28 janvier 2019 à 08:53
▶	folder	imgui-sfml	28 janvier 2019 à 16:31
▼	folder	io	aujourd'hui à 09:27
	file	cstr_file_generated.txt	aujourd'hui à 09:27
	file	cstr_file.txt	29 janvier 2019 à 11:26
▶	folder	misc	25 janvier 2019 à 13:48
▶	folder	src	aujourd'hui à 09:27
	file	CMakeLists.txt	7 février 2019 à 16:20
	file	topologyGUI	aujourd'hui à 09:27
	file	README.md	31 janvier 2019 à 15:27
	file	imgui.ini	aujourd'hui à 09:27

6 Les fichiers de contrainte simplifiés

TopologyGUI permet la génération de fichiers simplifiés. Ces fichiers simplifiés contiennent uniquement les informations essentielles d'un fichier classique (position d'un bloc avec ses coordonnées) et sont donc plus lisibles. Cependant, **ces fichiers simplifiés ne sont pas pris en compte par le logiciel Xilinx Vivado**.

Exemple de fichier de contrainte simplifié :

```
pblock_1 (GEN_Y[0] GEN_X[0])
X0 : 6 Y0 : 15 || X1 : 16 Y1 : 37

pblock_2 (GEN_Y[0] GEN_X[1])
X0 : 6 Y0 : 101 || X1 : 16 Y1 : 123

pblock_3 (GEN_Y[0] GEN_X[2])
X0 : 6 Y0 : 187 || X1 : 16 Y1 : 209

pblock_4 (GEN_Y[1] GEN_X[0])
X0 : 92 Y0 : 64 || X1 : 102 Y1 : 86

pblock_5 (GEN_Y[1] GEN_X[1])
X0 : 92 Y0 : 150 || X1 : 102 Y1 : 172

pblock_6 (GEN_Y[1] GEN_X[2])
X0 : 92 Y0 : 236 || X1 : 102 Y1 : 258

pblock_7 (GEN_Y[2] GEN_X[0])
X0 : 178 Y0 : 15 || X1 : 188 Y1 : 37

pblock_8 (GEN_Y[2] GEN_X[1])
X0 : 178 Y0 : 101 || X1 : 188 Y1 : 123

pblock_9 (GEN_Y[2] GEN_X[2])
X0 : 178 Y0 : 187 || X1 : 188 Y1 : 209
```

6.1 Le fichier simplifié original

1. Cliquez sur « **Generate simplified constraint file** » pour générer un fichier de contrainte simplifié du fichier original, sans aucune modification.

Generate simplified constraint file without any modification

Generate simplified constraint file (?)

6.2 Le fichier simplifié

1. Cliquez sur « **Generate constraint file (simplified)** » pour générer un fichier de contrainte simplifié suivant les caractéristiques sélectionnées auparavant (topologie).

Save constraint file as

.txt

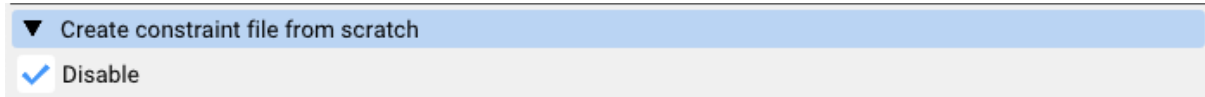
Generate constraint file

Generate constraint file (simplified) (?)

7 Comment modifier d'un fichier de contrainte ?

Il est possible de modifier les caractéristiques d'un fichier de contrainte en appliquant une certaine topologie plus homogène à celui-ci en utilisant la section « **Topology** ».


1. Vérifier que la case « **Disable** » de la section « **Create constraint file from scratch** » est bien cochée.



▼ Create constraint file from scratch

☒ Disable

2. Optionnel : Il est possible de redimensionner le premier bloc (et ainsi **tous les autres blocs**) du fichier de contrainte importé dans la section « **Dimensions** ». Saisissez la hauteur et la largeur du premier bloc.



▼ Dimensions

☐ Use default size ☒ Resize every pblock

Width : Height :

10 10

Vous avez bien sûr la possibilité d'effectuer aucune modification en sélectionnant l'option « **Default** » de la section « **Dimensions** ». Les dimensions du premier bloc sont affichées.

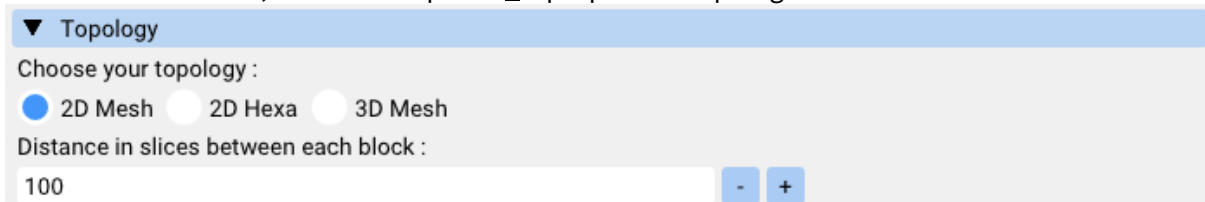


▼ Dimensions

☒ Use default size ☐ Resize every pblock

(10 , 23)

3. Choisissez la topologie que vous voulez appliquer au fichier de contrainte initial en saisissant la distance **d** ou **r**. Pour rappel, la distance **d** est la distance entre le centre de chaque bloc et la distance **r** est la diagonale entre les blocs hexagonaux. Pour plus d'indications, consultez la partie 3 à propos des topologies.



▼ Topology

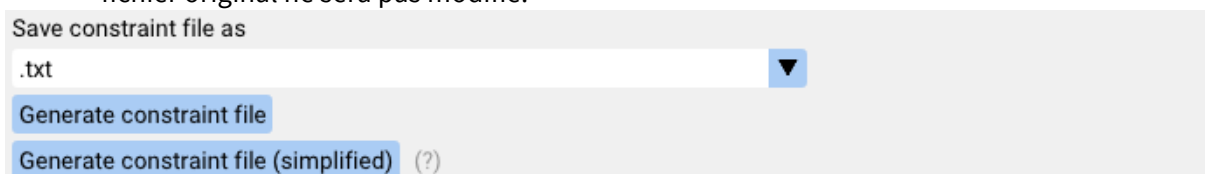
Choose your topology :

☒ 2D Mesh ☐ 2D Hexa ☐ 3D Mesh

Distance in slices between each block :

100 - +

4. Enregistrez le fichier de contrainte normalisé ou simplifié au format texte **.txt** ou format **.xdc** en cliquant sur « **Generate constraint file** » ou « **Generate constraint file (simplified)** ». Le fichier original ne sera pas modifié.



Save constraint file as

.txt ▼

Generate constraint file

Generate constraint file (simplified) (?)

8 Comment créer un fichier de contrainte ?

Il est possible de créer un fichier de contrainte (en ignorant le fichier de contrainte importé) en utilisant la section « **Create constraint file from scratch** ».

8.1 Création de topologies 2D et 3D

1. Décochez la case « **Disable** ». En décochant cette case, le fichier de contrainte initial sera ignoré.

▼ Create constraint file from scratch

☐ Disable

DimX :		DimY :		
3		3		9 blocs
X0 :	Y0 :	X1 :	Y1 :	
10	10	20	20	(W : 10, H : 10)

2. Remplissez les caractéristiques du fichier de contrainte de sortie en spécifiant le nombre de routeur en X (**dimX**), le nombre de routeur en Y (**dimY**) et les **coordonnées** du bloc1. Pour plus d'indications, consultez la partie 0 à propos des coordonnées des blocs.
3. Choisissez la topologie à appliquer (**2D maillée**, **2D hexagonale**, **3D maillée**) et spécifiez la distance **d** ou **r** que vous souhaitez utiliser. Pour plus d'indications, consultez la partie 3 à propos des topologies.

▼ Topology

Choose your topology :

☒ 2D Mesh ☐ 2D Hexa ☐ 3D Mesh

Distance in slices between each block :

100 - +

4. Enregistrez le fichier de contrainte normalisé ou simplifié au format texte **.txt** ou format **.xdc** en cliquant sur « **Generate constraint file** » ou « **Generate constraint file (simplified)** ».

Save constraint file as

.txt ▼

Generate constraint file

Generate constraint file (simplified) (?)

8.2 Remarques à propos de la topologie 3D

La topologie 3D maillée ne possède qu'une seule dimension Z, c'est-à-dire que le paramètre **dimZ** est fixé à **1** et ne peut être changé.

☐ Disable

DimX :	DimY :	DimZ : (always 1) (?)	
3	3	1	18 blocs

9 Comment gérer les surfaces d'une topologie ?

Il est possible de calculer la surface d'une topologie en utilisant la section « **Surface** ».

9.1 Comment calculer la surface d'une topologie ?

1. Cliquez sur le bouton « **Generate constraint file** ». La surface calculée est basée sur les paramètres des sections précédentes. Vous pouvez soit calculer la surface d'une topologie basée sur un fichier de contrainte importé ou soit calculer la surface d'une topologie créée par le logiciel en prenant soin de décocher la case « **Disable** ». Les surfaces s'affichent de la façon suivante :

Surface 2D (slices) : 0
Surface Hexa (slices) : 0
Surface 3D (slices) : 0

9.2 Comment calculer la distance à partir d'une surface ?

Dans la section « **Surface** », il est possible de calculer la distance entre le centre de chaque bloc d'une topologie en spécifiant la surface de celle-ci. Pour pouvoir utiliser ce mode :

1. Spécifiez la surface à partir de laquelle vous voulez calculer une distance.

Input surface S : 50000 - + (?)

2. Remplissez les caractéristiques du premier bloc de la topologie en utilisant la section en spécifiant le nombre de routeur en X (**dimX**), le nombre de routeur en Y (**dimY**) et les **coordonnées** du bloc1. Pour plus d'indications, consultez la partie 0 à propos des coordonnées des blocs.

▼ Create constraint file from scratch

☐ Disable

DimX :	DimY :	
3	3	9 blocs
X0 :	Y0 :	X1 : Y1 :
10	10	20 20 (W : 10, H : 10)

3. Appuyez sur le bouton « **Get Distance** » pour voir afficher le résultat, respectif à chaque topologie.

Input surface S : 50000 - + (?)

☐ Distance 'd' from surface S (2D Mesh) : 106
☐ Distance 'r' from surface S (2D Hexa) : 114
☐ Distance 'd' from surface S (3D Mesh) : 85

Get Distance **Apply selected distance**

4. Vous pouvez appliquer la distance **d** ou **r** calculée en sélectionnant la distance propre à la topologie souhaitée et en cliquant sur « **Apply selected distance** ». Le résultat apparaîtra ainsi dans la section « **Topology** ».

10 Comment récupérer un fichier de contrainte généré ?

Les fichiers de contrainte (normalisés ou simplifiés) sont générés dans le dossier **io** du projet au format imposé. Les fichiers générés respectent les appellations suivantes :

Topologie	Fichier généré (output) :	Fichier simplifié généré (output)
2D maillée	cstr_file_2D_generated.txt	cstr_file_2D_simplified_generated.txt
2D hexagonale	cstr_file_hexa_generated.txt	cstr_file_hexa_simplified_generated.txt
3D maillée	cstr_file_3D_generated.txt	cstr_file_3D_simplified_generated.txt

Exemple sous MacOS :

▼ topologyGUI	aujourd'hui à 09:27
▶ cmake_modules	3 décembre 2018 à 16:23
▶ cmake-build-debug	8 février 2019 à 09:51
▶ cmake-build-release	8 février 2019 à 09:51
▶ imgui	28 janvier 2019 à 08:53
▶ imgui-sfml	28 janvier 2019 à 16:31
▼ io	aujourd'hui à 09:27
cstr_file_generated.txt ←	aujourd'hui à 09:27
cstr_file.txt	29 janvier 2019 à 11:26
▶ misc	25 janvier 2019 à 13:48
▶ src	aujourd'hui à 09:27
CMakeLists.txt	7 février 2019 à 16:20
topologyGUI	aujourd'hui à 09:27
README.md	31 janvier 2019 à 15:27
imgui.ini	aujourd'hui à 09:27

11 Logs

Le logiciel possède une partie log permettant de vous informer sur les différentes opérations effectuées.

Log :	Clear	Copy	Filter
08-02-2019.11:22:25 [info] [surface] Calculate distances from surface S			
08-02-2019.12:17:36 [info] [mesh2D] Resize every blocs to (10,10)			
08-02-2019.12:17:36 [info] [mesh2D] Input file ignored			
08-02-2019.12:17:36 [info] [mesh2D] Generate constraint file			
08-02-2019.12:17:36 [info] [mesh2D] Success			
08-02-2019.12:17:36 [info] [mesh2D] Output file : fichier_contrainte_2D_generated.txt			

12 Bugs

Il se peut que le programme possède des bugs ou des erreurs mémoires. S'il-vous-plaît, envoyez un mail à l'adresse suivante ou de proposer une issue sur **Github** afin de les corriger : cedric.gormond@gmail.com