v8 du 20 jan & 27 fév 2023 / litzis@bluewin.ch

Principaux objectifs :

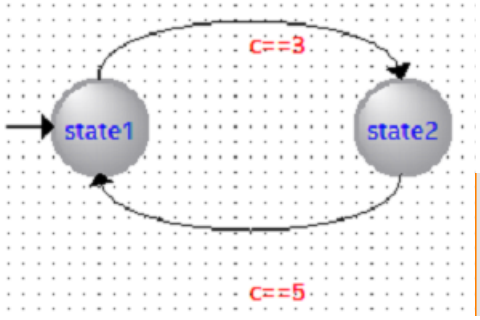
- Réaliser mon premier développement avec une **FPGA**
- **Eviter** dans la mesure du possible d'utiliser les langages Verilog ou VHDL
- Continuer d'utiliser les excellents produits **Arduino** avec la carte **Vidor 4000**
- Marier schéma logique & **machine d'état**
- Produire ce document et l'archive https://gelit.ch/Vidor/01.zip

Intérêt d'utiliser une machine d'état :

Il s'agit d'une **méthode essentielle en télécommunications pour structurer au mieux le logiciel**
Je l'utilise depuis ma découverte des réseaux de Pétri
https://fr.wikipedia.org/wiki/R%C3%A9seau_de_Petri
qu'il s'agisse de code de bas niveau comme l'émission du signal MFX (destiné aux décodeurs de mes locomotives voir lignes 186 à 241 de https://gelit.ch/Train/DirectMFX.ino) ou de l'implémentation d'un protocole de messagerie comme SMTP (lignes 287 à 334 dans https://gelit.ch/MKR/alarm.ino)

Extrait de https://gelit.ch/Train/DirectMM2.pdf page 6 :
Les personnes chargées de la spécification des protocoles utilisent ou ont utilisé le langage naturel (texte en anglais, ...) compréhensible par tous.
*Le monde internet regorge de ce type de **mauvaise spécification** :*
https://www.rfc-editor.org/rfc/rfc821
La problématique devient évidente lorsque 2 développeurs comprennent et implémentent la même spécification ambiguë de 2 manières différentes !
Ci-dessous un exemple de précieux document utilisant un formalisme simple et puissant dès la page 18
https://www.etsi.org/deliver/etsi_i_ets/300100_300199/30010202/01_60/ets_30010202e01p.pdf
Nous (mes étudiants lors de leur travail de diplôme et moi-même) avons vite compris les nombreux avantages de ce type de spécification !

L'exemple proposé de machine d'état se veut simpliste avec seulement 2 états :

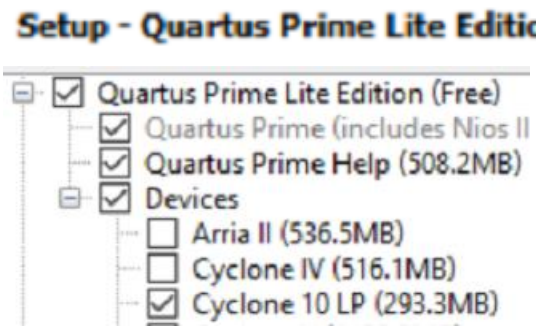



```
switch (SM) {
  case 1 : if (c==3) {SM=2;} break;
  case 2 : if (c==5) {SM=1;} break;
}
```

Mon apprentissage :

Je tiens à remercier **Philippe** pour la qualité des documents :
1. https://systemes-embarques.fr/wp/archives/arduino-mkr-vidor-4000-presentation-et-mise-en-route/
2. https://systemes-embarques.fr/wp/archives/mkr-vidor-4000-programmation-du-fpga-partie-1/

Le **premier** m'a facilité l'installation du logiciel **gratuit** Intel **Quartus Prime Lite Edition 22**
https://www.intel.com/content/www/us/en/software-kit/757262/intel-quartus-prime-lite-edition-design-software-version-22-1-for-windows.html après avoir décompressé le fichier .tar

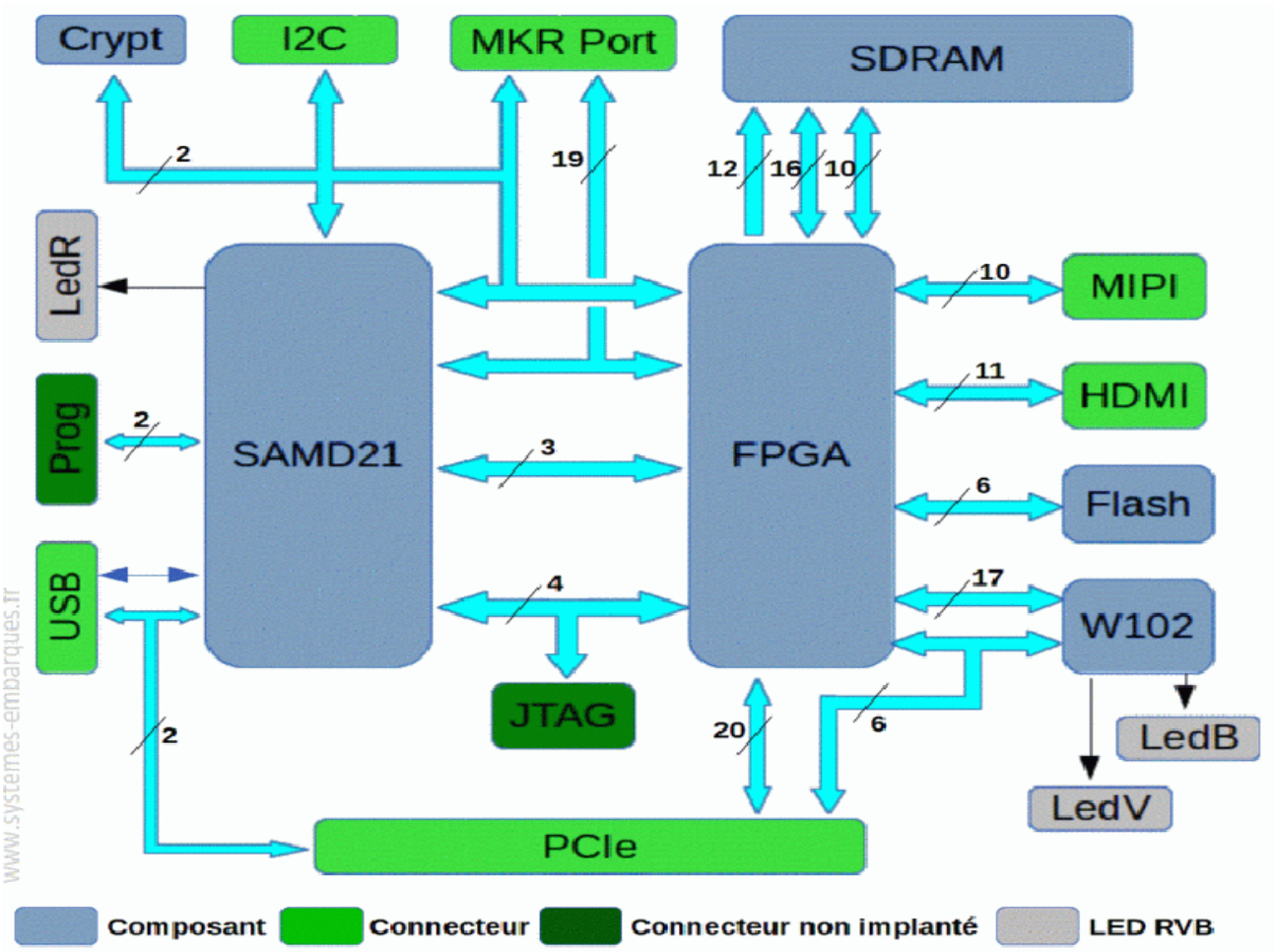


Le FPGA présent sur Arduino Vidor est le composant Intel Cyclone 10 LP
https://www.intel.com/content/www/us/en/docs/programmable/683879/current/device-overview.html

Le **second** document m'a permis de découvrir ce puissant outil graphique et de comprendre la construction du Lego FPGA

De plus, la constellation de cette carte Arduino Vidor semble irréelle
... ce n'est plus un couteau suisse mais ...

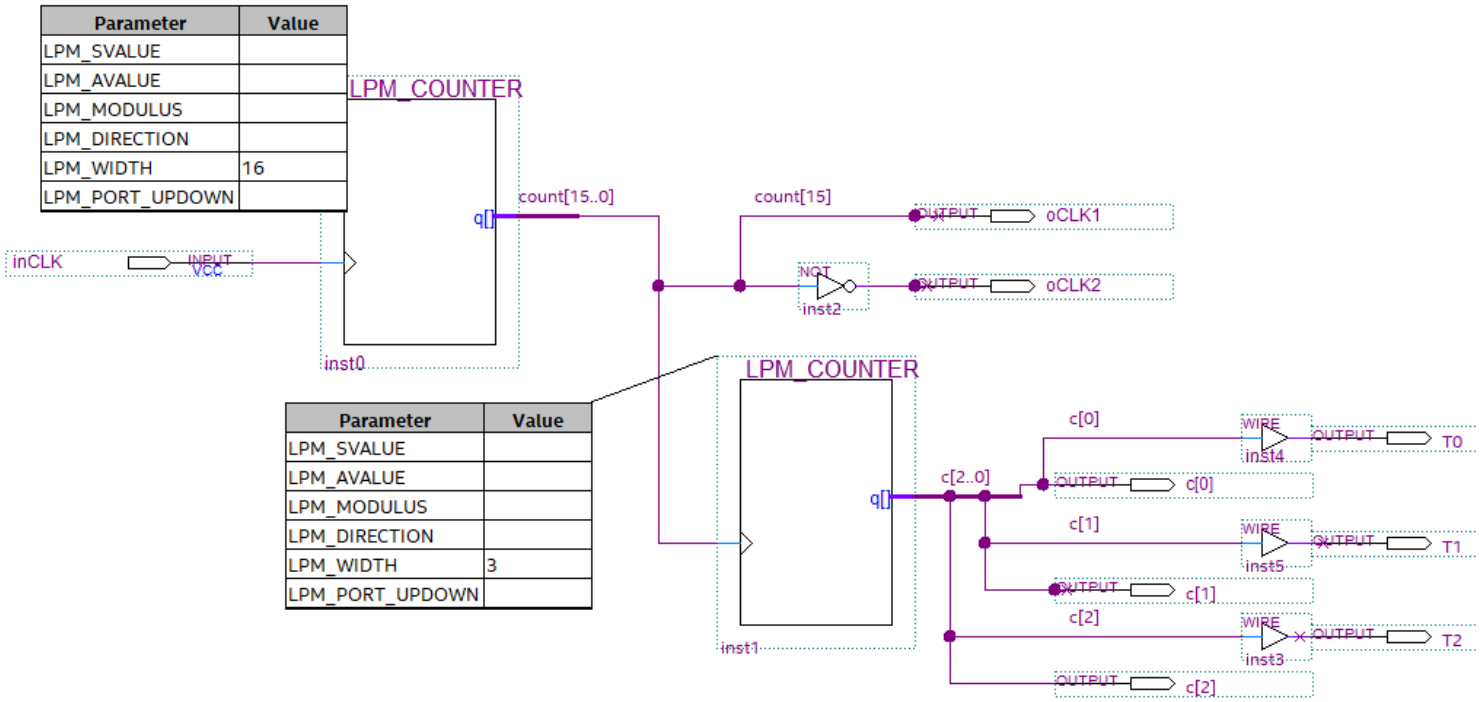
- un **microcontrôleur 32 bit cadencé à 48 MHz** embarquant **256kB de Flash** et **32kB de RAM**
- un **FPGA Cyclone 10CL016** avec 15408 éléments logiques, 504kbits de RAM et 56 multiplicateur 18×18.
- une FLASH SPI de 16 Mbits.
- une SDRAM de 64 Mbits (4M x 16 bits)
- un excellent module **Wifi/BLE NINA W102**
- des **connecteurs MiniPCIe, USB, batterie, I2C, MKR, MIPI pour une caméra, HDMI pour la sortie vidéo**



Objectif de ce projet :

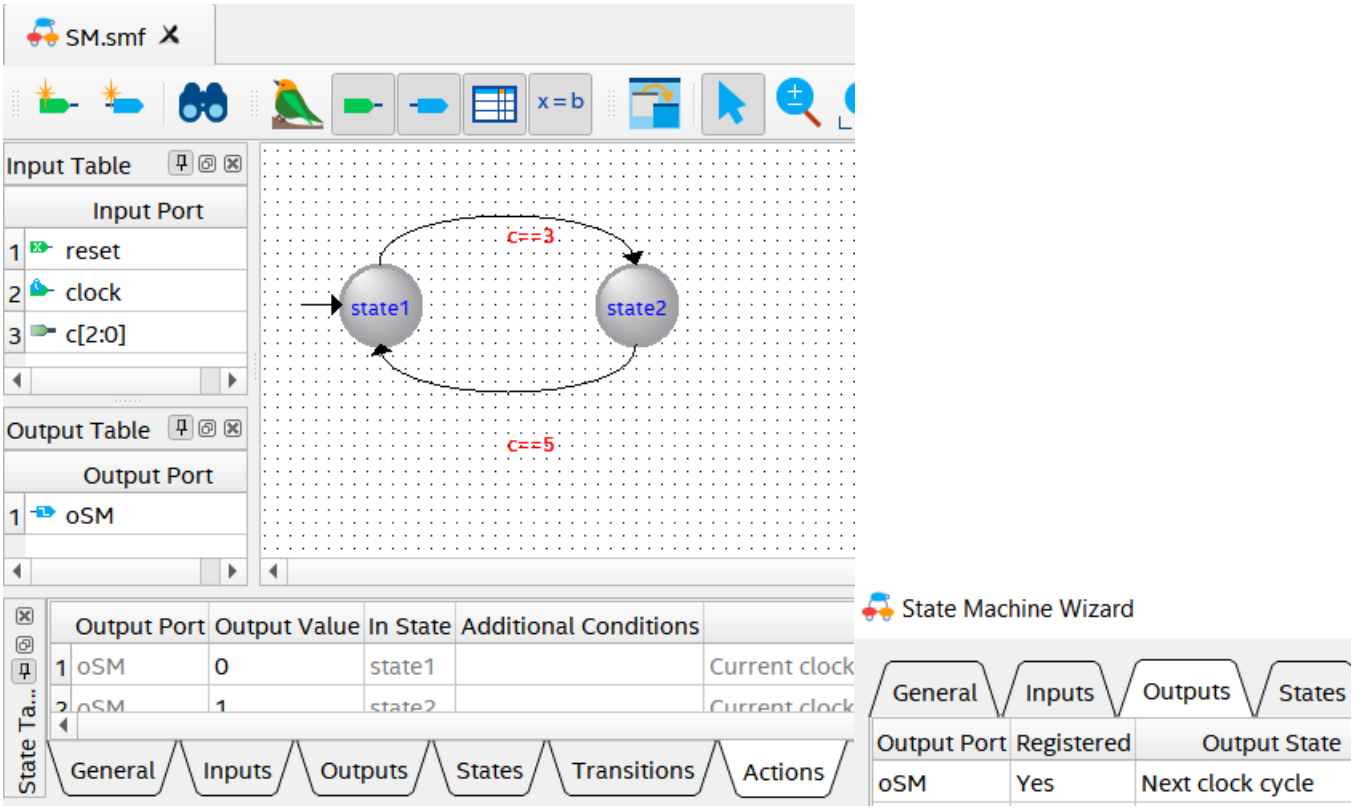
Utiliser un compteur (modulo 8) dont les 3 sorties c[2..0] seront évaluées par la machine d'état

Schéma



Machine d'état

selon SM.v : *always @(posedge clock)* signifie transition sur le flanc positif; ce qui explique l'inverseur inst2 afin que la machine d'état utilise les états stables du LPM_COUNTER (qui change d'état sur le flanc montant de oCLK1)



_top.v
ce fichier est essentiel pour comprendre :

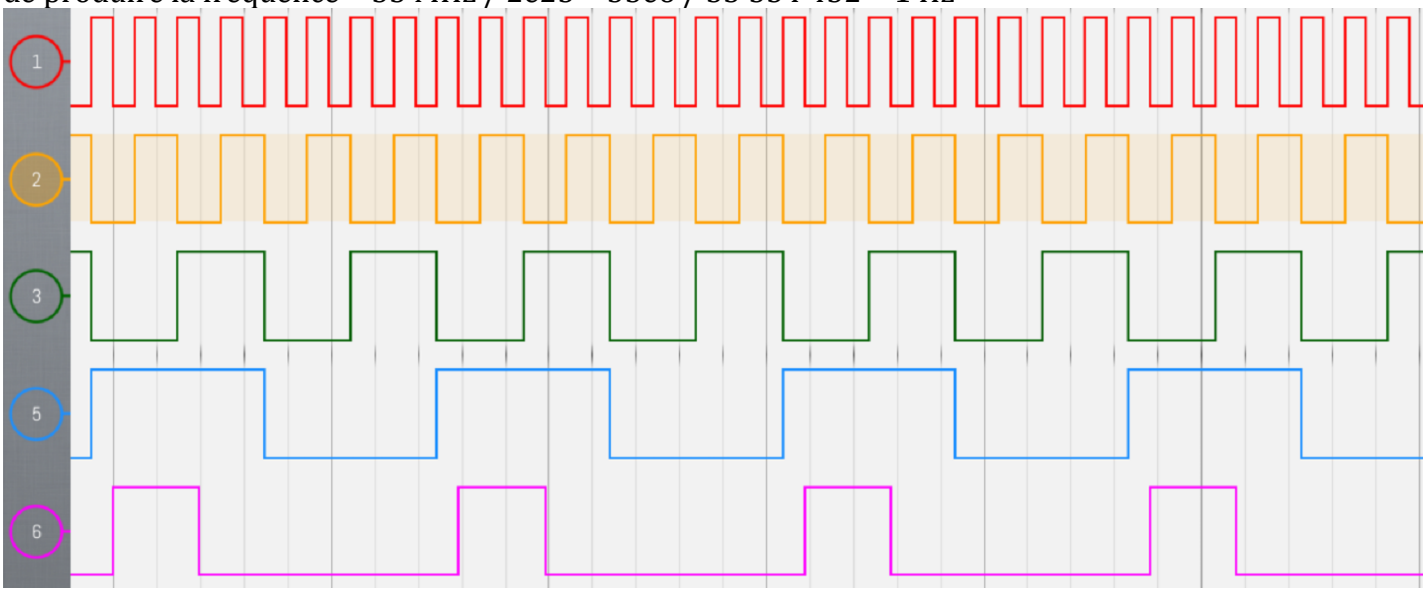
- **les connexions physiques entre modules**
- **les connexions avec le connecteur MKR**

```
wire [2:0]  data;  //                              connected with data
wire        ww;    //                    connected with ww

Schema Schema_inst
(
.inCLK(wOSC_CLK)  , // 33 MHz
.oCLK1(bMKR_D[6]), // Clock MKR pin6 = 33/2e15 = 1 MHz
.oCLK2(ww),        // !Clock to SM      connected with ww
.T0(bMKR_D[5]),    // test purpose
.T1(bMKR_D[4]),    // idem
.T2(bMKR_D[3]),    // idem
.c(data)           //                              connected with data
);
SM SM_inst
(
.clock(ww),        //                    connected with ww
.oSM(bMKR_D[7]),   // SM output MKR pin7
.c(data)           //                              connected with data
);
```

Analyseur logique

Les personnes dépourvues d'analyseur logique peuvent remplacer le signal count[15] par count[25] comme dans https://systemes-embarques.fr/wp/archives/mkr-vidor-4000-programmation-du-fpga-partie-1/ afin de produire la fréquence = 33 MHz / 2e25 = 33e6 / 33 554 432 = 1 Hz



- Canal 1 = oCLK1 (clock des 2 compteurs)
- Canal 2 = T0 (compteur change d'état sur le flanc positif de oCLK1)
- Canal 3 = T1
- Canal 5 = T2
- Canal 6 = oSM (output SM avec transition sur flanc négatif de oCLK1)
  = 1 si c[2:0] = T[2:0]==3
  0 5

Marche à suivre :

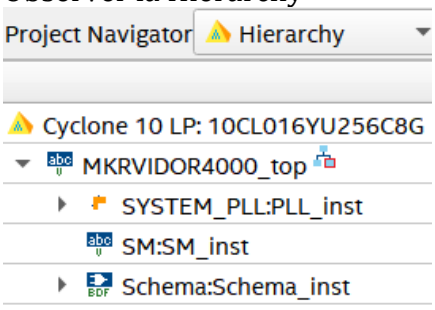
1. Décompresser puis copier l'arborescence **dans le dossier ...\Vidor\01**

2. Dans ...\Vidor\01, exécuter MKRVIDOR4000.QPF pour démarrer Quartus dans le bon dossier **Contrôler** en haut de le fenêtre

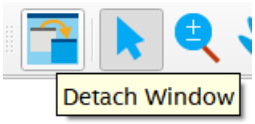
   Quartus Prime Lite Edition - D:/Vidor/01/

3. Upgrade si demandé

4. Observer la Hierarchy



5. Double-clic sur Schema que vous pouvez détacher

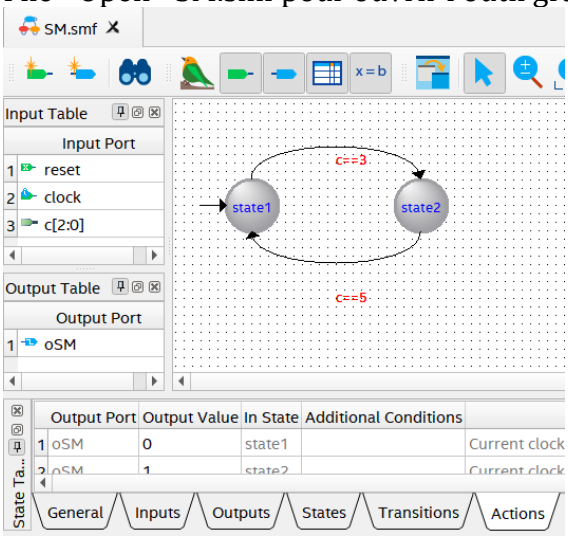


6. Double-clic sur SM

```
21  `timescale 1ns/1ns
22
23  module SM (
24      reset,clock,c[2:0],
25      oSM);
26
27      input reset;
28      input clock;
29      input [2:0] c;
```

Une extension .v (=Verilog) s'ouvre que l'on peut heureusement ignorer !!!

7. File - Open - SM.smf pour ouvrir l'outil graphique

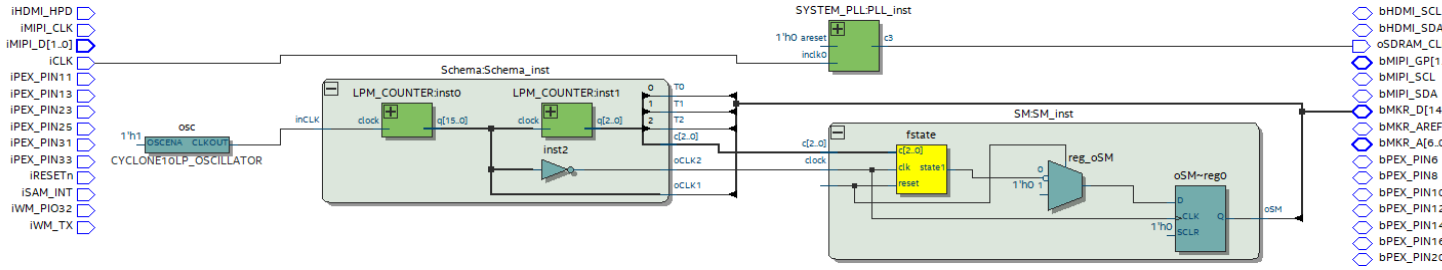


8. Compiler avec la flèche bleue

Vous savez que la compilation est terminée en observant STOP
Contrôler

```
Quartus Prime Timing Analyzer was successful. 0 errors,
293000 Quartus Prime Full Compilation was successful. 0 errors
```

9. **Le plus important en cas de problème**
Tools - Netlist Viewers - RTL Viewer



J'ai perdu beaucoup de temps avant de comprendre que les connexions c[2..0] étaient absentes entre le module Schema et le module SM !!!
Solution dans Hierarchy - MKRVIDOR4000_top en ligne 161

```
161    wire [2:0]  data;
```

Les blocs bleus dans la machine d'état correspondent aux options Registered & Output States

State Machine Wizard

General	Inputs	Outputs	States

Output Port	Registered	Output State
oSM	Yes	Next clock cycle

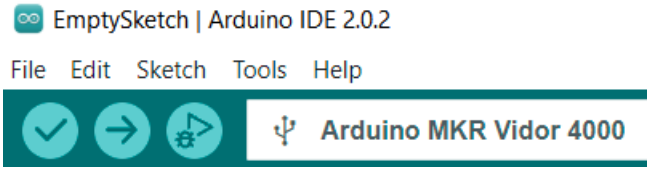
afin que la sortie traverse un flip-flop dont l'état change avec un cycle de décalage que l'on peut observer sur les mesures faites avec l'analyseur logique

10. Installer Java

11. Modifier le fichier app.h
Dans ...\Vidor\01\VidorFPGA-master\projects\MKRVIDOR4000_template\output_files
Shift + clic-droit - Open PowerShell

```
\Vidor\01\VidorFPGA-master\projects\MKRVIDOR4000_template\output_files> java ReverseByte MKRVIDOR4000.ttf app.h
```

12. Installer Arduino IDE 2.0
https://docs.arduino.cc/software/ide-v2/tutorials/getting-started/ide-v2-downloading-and-installing
J'utilise un PC Windows 10 64 bit
Exécuter **...\Vidor\01**\EmptySketch\EmptySketch.ino
Attention à sélectionner la bonne carte Arduino



Contrôler que le dossier est bien le bon à la ligne 54

```
54    #include "D:\Vidor\01\
```

Upload EmptySketch

```
done in 6.861 seconds
CPU reset.
```

13. Brancher votre analyseur logique aux pins de la carte Vidor

14. Je conseille, **pour votre prochain projet**, de conserver les fichiers dans le dossier ...\Vidor\01 puis de créer le dossier ...\Vidor\02 dans lequel vous allez copier tous les fichiers du dossier ...\Vidor\01

Vous pourrez ainsi facilement revenir en arrière

Principales difficultés rencontrées :

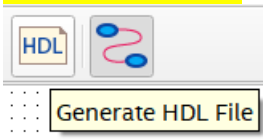
1. Contrôler régulièrement que vos outils (Quartus, PowerShell, IDE) accèdent au bon dossier

2. la ligne 161 ci-dessus pour comprendre le rôle de l'outil RTL Viewer (qui nous montre les entrailles de la FPGA)

3. La compilation ne prend pas en compte les modifications que vous faites dans une machine d'état avec l'outil graphique

   ⚠ 12125 Using design file sm.v, which is not specified as a design file for the current project, but contains definitions for 1 design units and 1 entities in project

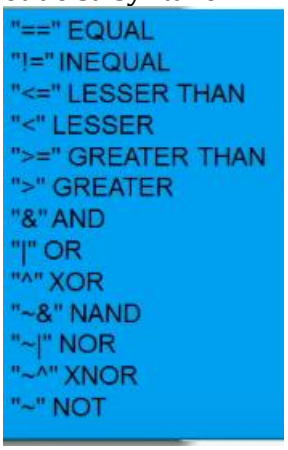
   Vous devez donc produire le fichier .v (Verilog) après chaque modification dans State Machine Wizard

4. Depuis Hierarchy, ouvrir MKRVIDOR4000_top pour identifier les **connexions** avec l'extérieur pour les modules et les branchements avec l'analyseur logique

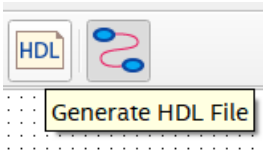
   Ce fichier, produit par Dario Pennisi (Arduino), se situe au sommet de la hiérarchie
   https://github.com/vidor-libraries/VidorFPGA

5. Découvrir l'usage de State Machine Wizard
https://www.youtube.com/watch?v=08YHxtSI3Bk
et de sa syntaxe



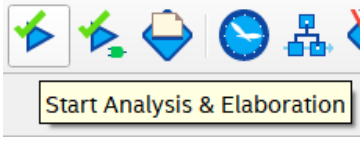
Répétitions importantes :
- Sauver vos modifications si l'onglet comporte *

- Ne jamais oublier de produire le fichier .v de cette machine d'état avec

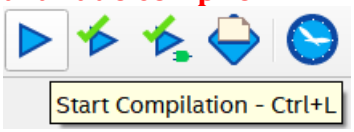


6. Parfois pas de compilation

J'ai pris l'habitude de commencer par Start Analysis & Elaboration



avant de compiler



Conclusion :

Les **outils graphiques Quartus** sont dans la philosophie des excellents produits Arduino destinés, en premier lieu, **à un public novice et curieux d'apprendre.**

Avec un minimum de connaissance du langage Verilog, j'ai pu facilement adapter la configuration matérielle désirée dans le fichier MKRVIDOR4000_top.v

Autres liens utiles :

Pour mon ami Stefano : quelques modules disponibles !!!
https://flex.phys.tohoku.ac.jp/riron/vhdl/up1/altera/cat/lpm.pdf

Pour ceux qui veulent comprendre le fonctionnement interne d'une FPGA
https://moodle.luniversitenumerique.fr/mod/page/view.php?id=1674

Pour ceux qui veulent retourner à l'école (209 pages)
https://www.montana.edu/blameres/book_content_vhdl/Lab_Exercises_LaMeres_Intro_to_Logic_wVHDL_DE10-Lite_May2022.pdf

A propos de Netlist Viewers
https://courses.cs.washington.edu/courses/cse467/08au/labs/Resources/Quartus%20II%20Netlist%20Viewers.pdf

Comment connecter les modules Verilog
https://www.youtube.com/watch?v=qgsEB7AbN0U

Brochage
https://systemes-embarques.fr/wp/brochage-connecteur-mkr-vidor-4000/

Propositions d'amélioration :

Pour une personne sans analyseur logique, il serait intéressant d'investiguer les possibilités de Quartus en matière de simulation
Est-il possible de simuler le signal Data ?
Peut-on imaginer utiliser des outils comme Simulation Waveform Editor ou ... ?

Prochain épisode : Analyseur logique du 10 fév 2023 compatible Windows 10 basé sur Arduino Vidor en https://gelit.ch/Vidor/LA.pdf