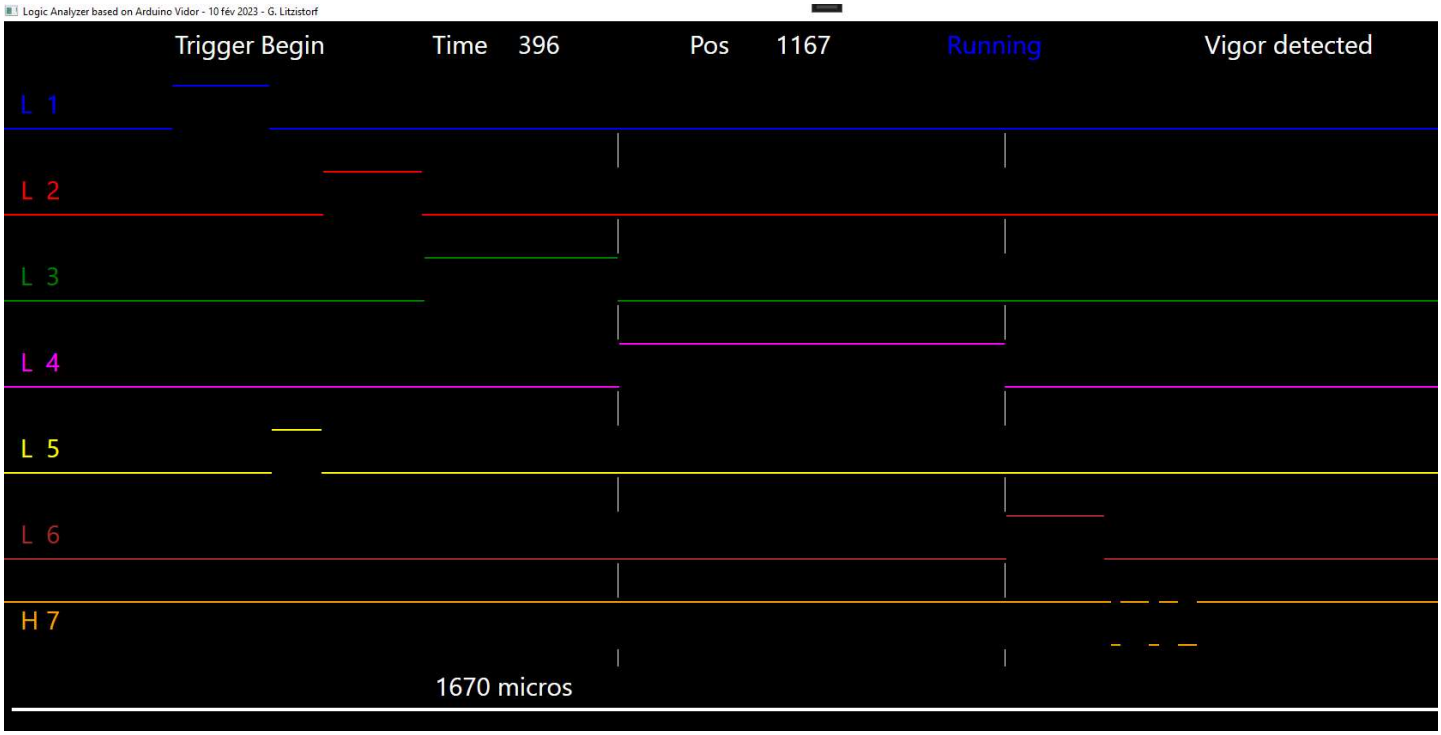
v5 du 13 fév 2023 / litzis@bluewin.ch

Principaux objectifs :

- Permettre à celui qui possède une carte **Arduino Vidor** d'avoir un **analyseur logique** pour un investissement inférieur à 100 CHF
- Démontrer qu'il est possible d'obtenir un résultat **sans connaissance du langage Verilog**
- Mettre à disposition un **squelette** pour l'environnement Microsoft **WPF** (Windows Presentation Foundation) avec ***dynamic DrawLine, PreviewKeyDown & Serial("COM", 800000)***
- Présenter quelques méthodes efficaces pour ***debugger*** un tel projet
- Remplacer mon analyseur LabNation doté d'un mécanisme de Trigger ennuyeux
- Fournir **https://github.com/gelit/Logic-Analyzer-with-Arduino-Vidor**

A quoi sert un analyseur logique ?

L'analyseur logique est l'outil essentiel pour **tester et valider des développements matériels**
Il permet **dans une fenêtre temporelle**, par exemple de 1670 µs (= micro-seconde), de visualiser plusieurs signaux numériques :



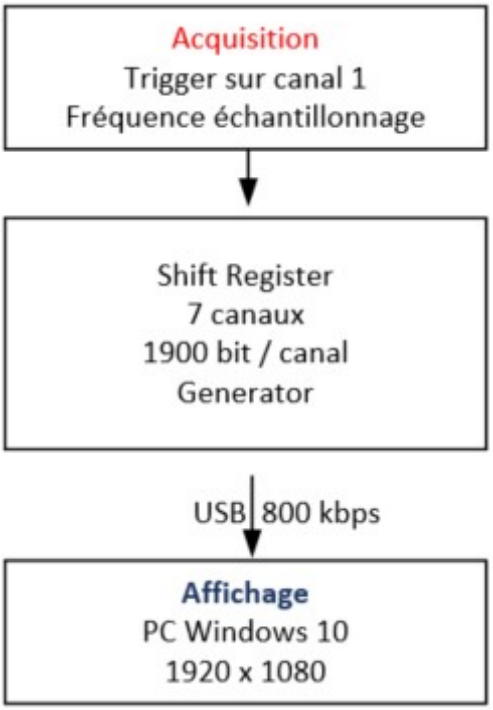
Cet exemple visualise 7 canaux (= signaux numériques)

L'analyseur logique ne connait que les états 0 et 1 ; inutile donc de tenter de mesurer un temps de montée
Contrairement à la grande majorité des analyseur, l'affichage proposé n'affiche donc aucun flanc !

Time = 396 indique l'interval de temps compris entre les 2 curseurs ; l'unité (µs ou ms) est précisé en bas avec 1670 µs

Pos donne la valeur en pixel du 1er curseau (utile pour vérifier que la résolution horizontale est bien égale à 1920 pixel)

Principe de fonctionnement



Durant la phase d'**acquisition**, les 7 signaux sont lus (échantillonnés) chaque 0.879 µs pour remplir une mémoire de 1900 bit
La fenêtre d'acquisition est donc = 1900 x 0.879 = 1670 **µs (microseconde)** = 1.67 **ms (milliseconde)**

La fréquence d'échantillonnage est, dans ce cas, égale à 1.14 MHz puisque F=1/T (Fréquence = inverse de la Période)

L'acquisition est démarrée *(Trigger)* sur le premier flanc montant *(positive edge)* ou descendant *(negative edge)* du signal (canal) 1

Au niveau de l'**affichage** :

- La résolution horizontale choisie (1920 pixel) associe un pixel pour chaque 0.879 µs (échantillon)
- La résolution verticale affiche 7 signaux (canaux)

Grâce à USB, il est imaginable de brancher N cartes pour disposer d'un appareil affichant N x 6 entrées + canal 1 commun (Trigger)

Utilisation simple & efficace

La non détection de la carte est signalée par un beep et une message explicite NO DETECTION ! à la place de Vigor detected

La première colonne affiche l'état de chaque entrée avec un rafraichissement de 1 seconde

Inutile donc d'espérer une acquisition si le canal 1 reste à L (Low) ou H (High)

L'affichage par défaut place le flanc du Trigger au pixel 200 ; soit à environ 10% des 1900 disponibles
Raccourcis clavier :

- **B** (Begin)
- **C** (Center) avec un flanc situé à 50%
- **E** (End) 90%

S (Stop) verrouille l'acquisition afin de conserver les données de la dernière acquisition

Etat indiqué par Stopped ou Running (mode par défaut)

La **fenêtre d'acquisition** est volontairement limitée à la largeur de l'écran (1900 pixel) et varie en fonction de la fréquence d'échantillonnage choisie avec les **flèches haut et bas :**

- Min = 104 µs
- Max = 107 ms

Les flèches horizontales déplacent les curseurs
La touche Tab sélectionne le curseau 1 ou 2
La touche LeftShift accélère le déplacement de 10
La position du curseur 1 est affichée en première ligne
L'interval de temps, compris entre les 2 curseurs, est affiché
Typer Z pour cacher les curseurs

Le logiciel intègre un générateur de signal avec sortie sur A0 dont les séquences sont accessibles avec les touches 1 à 9

Il offre divers choix (facilement modifiables) dans le Sketch Arduino

```
case '1' : digitalWrite(A0,1); delayMicroseconds(100); digitalWrite(A0,0);
break;

case '2' : for (N=1; N<=14; N++) {digitalWrite(A0,1); delayMicroseconds(100);
                                  digitalWrite(A0,0); delayMicroseconds(100);}
break;
```

Limitations : la carte dispose de 22 pins qui sont toutes utilisées

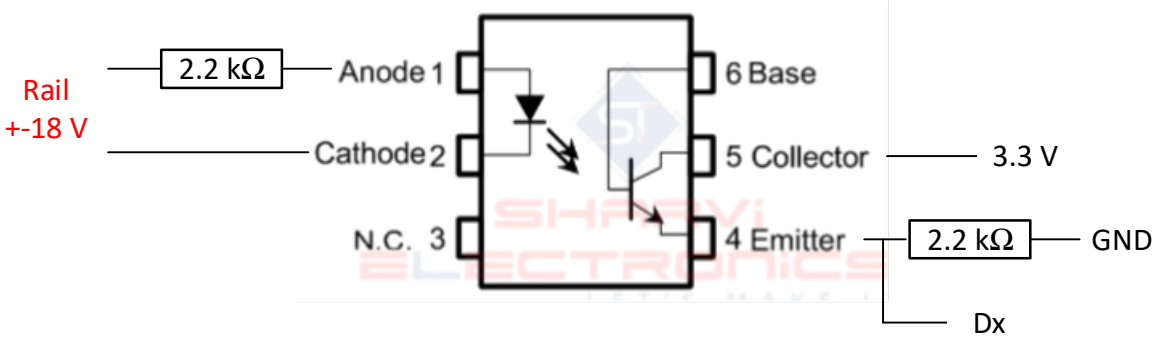
	A disposition	Usage interne
• D0		Tshift (FPGA → CPU)
• D1-D7	7 input	
• D8-D14		7 output (FPGA → CPU)
• A0	Generator	
• A1		Byte Available (FPGA → CPU)
• A2-A4		Féch Fréquence échantillonnage (FPGA ← CPU)
• A5-A6		Mode du Trigger (FPGA ← CPU)

Attention – Danger !!!
La personne qui utilise cet analyseur logique doit comprendre les caractéristiques (limites) d'une entrée Dx
Selon
https://www.intel.com/content/www/us/en/docs/programmable/683251/current/recommended-operating-conditions.html

V_I	Input voltage	—	–0.5	—	3.6	V

La tension max est limitée à 3.6 V

Lors de mes mesures sur le rail Märklin, avec une tension alternée +18V- avec -18V+, j'utilise un optocoupleur CNY17 afin de limiter la tension à 3.3 V et d'assurer une séparartion galvanique



Choix

Ils facilitent l'utilisation de cet analyseur logique :

- Un seul Trigger possible avec le canal 1 sur le flanc montant ou descendant
- Toute la mémoire des 7 canaux de 1900 bit est affichée
- **Fenêtre d'acquisition**

Fenêtre d'acquisition		Tech		Féch			
µs	ms	ns	ms	MHz	kHz		
104		55		18.2		Clk[0]	
209		110		9.1		Clk[1]	
418		220		4.55		Clk[2]	
835		439		2.28		Clk[3]	
1670		**879**		**1.14**		Clk[4]	**Valeur par défaut**
	3	1758	1.76		569	Clk[5]	
	7		3.51		284	Clk[6]	
	13		7.03		142	Clk[7]	
	27		14.1		71	Clk[8]	
	53		28.1		35.5	Clk[9]	
	107		56.3		17.8	Clk[10]	

Mesures effectuées avec la précision du LabNation

- 2 curseurs pour mesurer un interval de temps
- Trigger en mode Begin – Center – End ou Stop
- Générateur de signal (sortie A0) facilement modifiable
- Raccourcis clavier → pas de souris

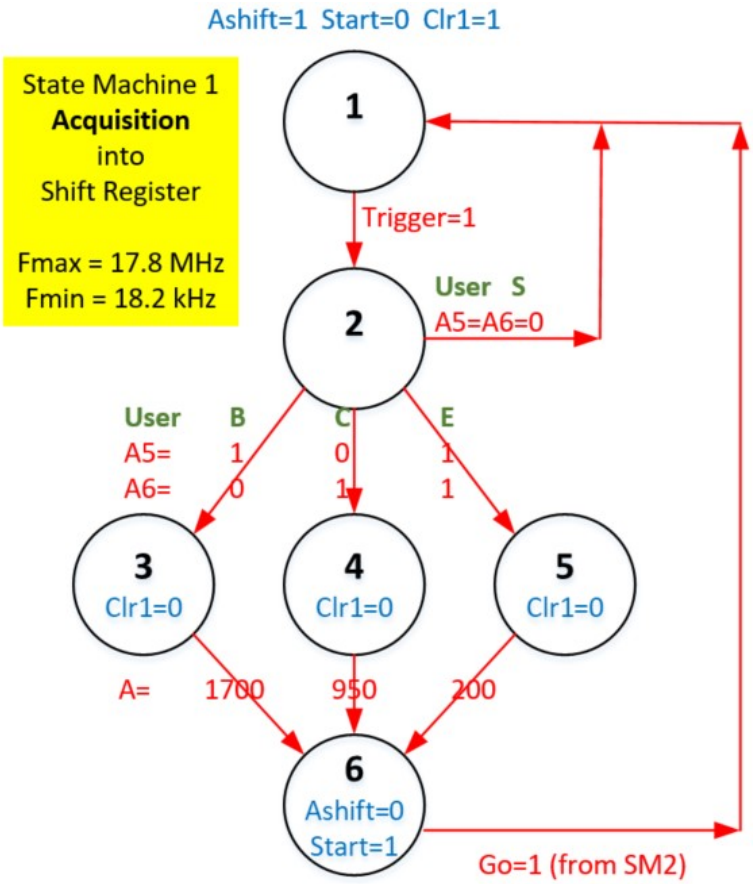
Modification de la fréquence d'échantillonnage

Il est essentiel que l'utilisateur comprenne que cette modification va entraîner un nouvel affichage !
Avec la flèche vers le bas, la fenêtre d'acquisition augmente (double) si bien que l'affichage prend fin à la moitié.
Par contre avec la flèche vers le haut, les 1900 pixels du nouvel affichage correspondent à la moitié des données acquises

Ce mécanisme est itératif tant que l'analyseur ne reçoit pas une nouvelle acquisition

Il est donc conseillé, après avoir modifié la fenêtre d'acquisition, d'effectuer une nouvelle acquisition afin d'afficher les 1900 pixels de chaque canal

2 machines d'état (avec des fréquences différentes) assurent le bon **séquencement**



La **1ère machine d'état (SM1)** gère l'acquisition des données des 7 canaux dans les registres à décalage.

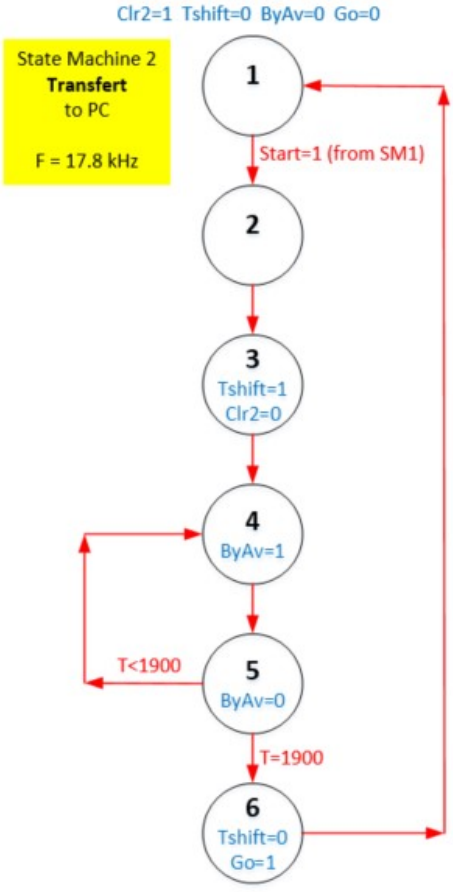
Dans l'état initial 1, Ashift=1 autorise l'acquisition

Elle passe dans l'état 2 au premier flanc positif ou négatif → Trigger=1

Les états 3 – 4 – 5 sont choisis en fonction du mode Begin – Center – End sélectionné

Le mode par défaut Begin affiche ce premier flanc du canal 1 à la position (pixel) 200 ; ce qui exige d'acquérir 1700 bit après la détection du Trigger

Ashift=0 indique la fin de l'acquisition
Start=1 démarre la 2ème machine d'état
SM1 attend sur Go=1 (généré par SM2)



La **2ème machine d'état (SM2)** gère le transfert des données vers le PC via USB

Il démarre avec Start=1 généré par SM1

Dans l'état 3, Tschift est activé pour autoriser le transfert (décalage)

L'état 4 avec ByAv (Byte Available) signale au CPU que les données sont disponibles

Dans ce mécanisme synchrone, CPU doit être capable **en 56 ms (F=17.8 kHz)** de lire les 7 canaux et d'envoyer les données compressés vers le PC

L'état 6 met fin au transfert et place SM1 dans son état initial

Format des données sur USB

Visible dans View – Output de VisualStudio grâce à `Debug.Write(Rx);`

Delta=3 Très utile lors du Debug pour contrôler le bon alignement → aucun bit perdu
Il suffit d'augmenter F pour comprendre les problèmes

1C	Canal 1	Clear	
1L200		Low	→ 200
1H313		High	→ 313
1L1903		Low	→ 1903
…			
7C	Canal 7 …		
7H1296			
…			
7L1903			

L'affichage de la première page comprend 7 canux de 1900 bit = 13'300 bit

Notepad++ indique une longueur totale = 220 caractères ; soit 2200 bit sur USB (10 bit / caract.)

Le facteur de compression = 1 - 2200/13300 = 83 %

Mes trucs pour *debugger* :

ToggleLed permet de tester si une procédure est appelée, le résultat d'un test ou … sans impacter la charge CPU

Serial1.print utilise les pins 13 – 14 et permet de rediriger la sortie USB vers un terminal putty

Afficher les états de la State Machine en affectant les sorties S0 – S1 – … avec les puissances de 2

Utiliser cet analyseur logique

Points d'amélioration :

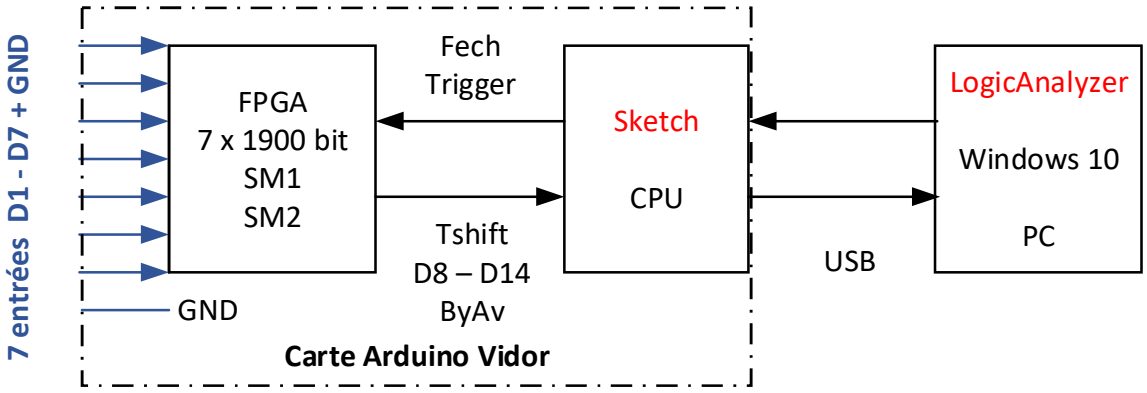
Ce projet se veut **pédagogique** ; il ne comporte donc pas toutes les fonctions d'un analyseur haut de gamme ; **laissant chacun ajouter ce qui lui manque**

Quelques idées qui m'intéressent :

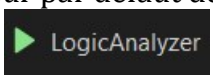
- Activer une PLL sophistiquée *(PLL synthetizer)* pour utiliser des fréquences d'échantillonnage plus élevées que 18 MHz ; par exemple 350 MHz !
  Clock Tree Max = 402 MHz pour 10CL016 … C8
  selon https://www.intel.com/content/www/us/en/docs/programmable/683251/current/clock-tree-specifications.html
  PLL output frequency (C8) = 402.5 MHz
  selon https://www.intel.com/content/www/us/en/docs/programmable/683251/current/pll-specifications.html

- Est-il possible de lire 8 *digital input* simultanément ?
  Mon vieux Motorola 68000 le permettait très facilement !!!

Vue d'ensemble

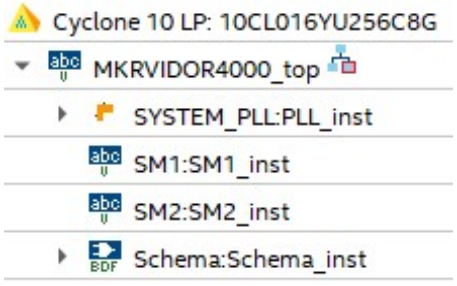


Mise en œuvre minimaliste (sans modification de la FPGA)

1. Depuis https://github.com/gelit/Logic-Analyzer-with-Arduino-Vidor Clic sur Code – Download ZIP
2. Mettre les fichiers décompressés dans le dossier …\Vidor\LA
3. Installer Arduino IDE 2.0 puis le lancer
4. Identifier le port USB COMx utilisé
5. Charger …\Vidor\LA \Sketch\Sketch.ino
6. Installer Visual Studio 2022
7. Project – Manage Nuget Packages pour mettre à jour System.IO.Ports
8. View – Solution Explorer
9. Double-clic sur MainWindow.xaml.cs
10. Ctrl F pour rechercher **COM**
11. Modifier la valeur par défaut de 3 en fonction du résultat obtenu en 4
12. Compiler avec 

Mise en œuvre pour étudier ou modifier la FPGA

1. Posséder les bases du logiciel Intel Quartus avec https://gelit.ch/Vidor/Vidor1.pdf
2. Installer puis lancer ce logiciel selon page 5 du document
3. Parcourir les 4 fichiers principaux de ce projet : _top, SM1, SM2, Schema



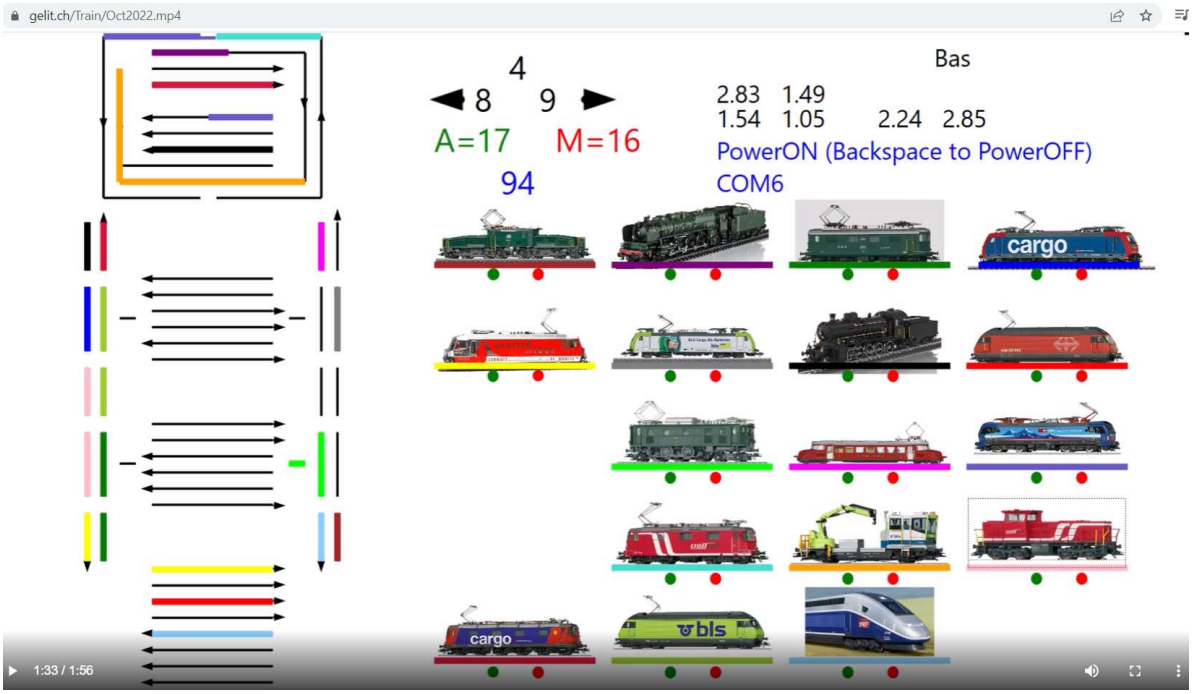
La technologie FPGA a révolutionné le développement matériel en supprimant stock de circuits intégrés et pistolet à wrapper
L'excellent outil Intel Quartus **gratuit** associé à des bibliothèques telles que https://flex.phys.tohoku.ac.jp/riron/vhdl/up1/altera/cat/lpm.pdf offre à la personne motivée par le développement matériel un magnifique terrain de jeu
De plus, en possédant 2 cartes Vidor, cette personne pourra développer sur la carte 1 et tester avec l'analyseur sur la carte 2

Mise en œuvre pour étudier ou modifier le programme Windows (suite de minimaliste)

- Dans Solution Explorer, double-clic sur MainWindow.xaml qui contient la définition **statique** des objets (Grid, Canvas, Text, Line) affichés
- MainWindow.xaml.cs contient le programme c#

A propos de MainWindow.xaml :

- La très grande majorité des exemples fonctionnels trouvés sur internet gère les objets graphiques WPF (Windows Presentation Foundation) de manières **statiques** !!!
- Heureusement que Google m'a aidé à trouver ce magnifique exemple https://www.youtube.com/watch?v=cvfkz0s6czA pour écrire **dynamiquement** une ligne !!!
- Mon implémentation actuelle du curseur évite de le superposer avec les 7 canaux affichés car je n'ai, pour l'instant, pas trouvé l'équivalent de zindex pour les objets statiques
- J'ai prévu pour 2023 d'utiliser la librairie graphique compatible HDMI de la carte Vidor https://www.youtube.com/watch?v=QSbFltEfQBs pour tester ses performances en remplaçant mon affichage Windows 10 WPF statique de ma maquette de trains



Pour plus d'info :
https://gelit.ch/Train/Video.mp4
https://gelit.ch/Train/H4.pdf

En conclusion je suis heureux d'annoncer, grâce au précieux outil Intel Quartus Lite, qu'il est possible de développer une solution FPGA sans connaissance du langage Verilog

La seule difficulté, facilement maîtrisable pour un adolescent, consiste à définir correctement entrée, sortie et connexion (wire) dans le fichier https://github.com/gelit/Logic-Analyzer-with-Arduino-Vidor/blob/main/VidorFPGA-master/projects/MKRVIDOR4000_template/MKRVIDOR4000_top.v

Intel Quartus Lite poursuit la philosophie des excellents produits Arduino en mariant simplicité, qualité et prix abordable

Intel reste pour moi le leader du matériel. Cette société, à la notoriété très élevée, devrait corriger certains bugs qu'un utilisateur lambda observe après 1 heure d'utilisation.

Je reste à disposition pour tout complément Gérald Litzistorf – prof. retraité – https://gelit.ch/