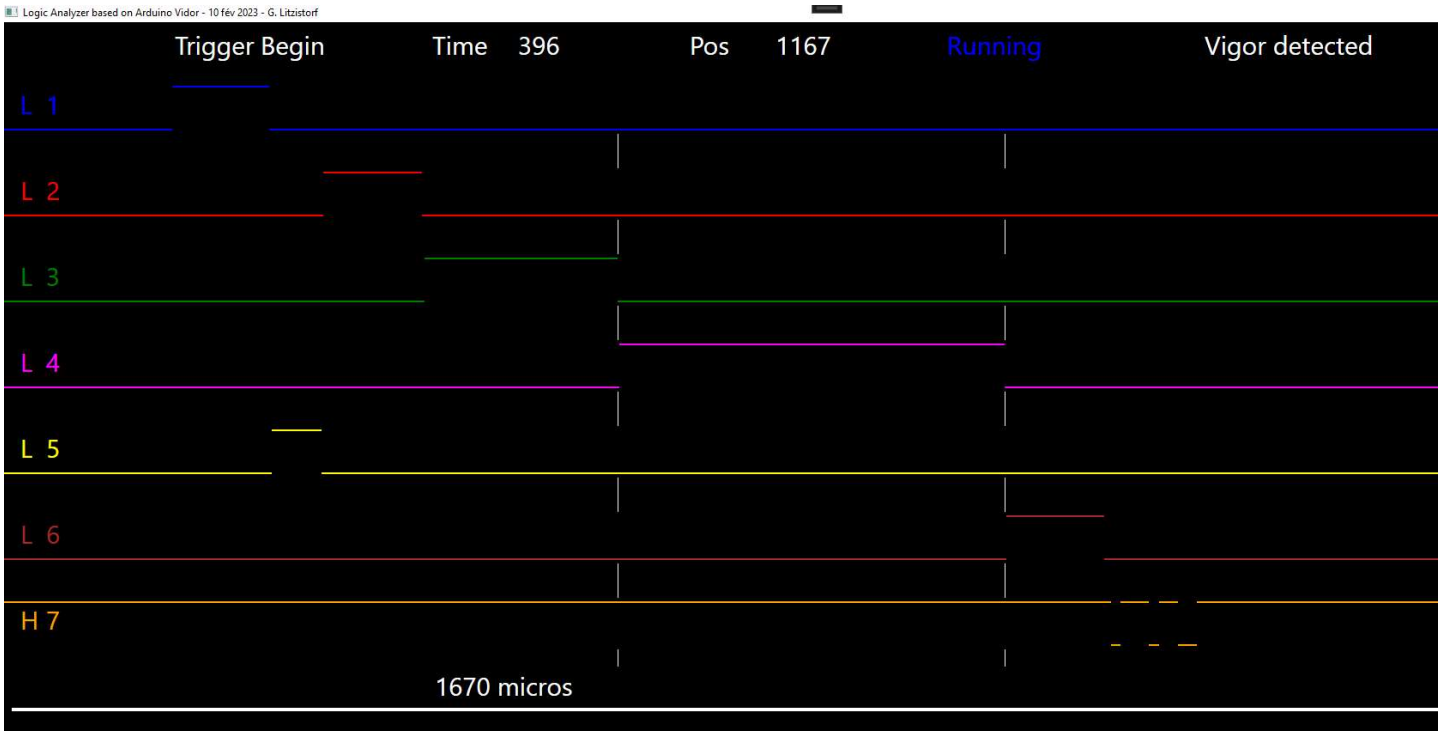
v3 du 13 fév 2023 / litzis@bluewin.ch it is a Google translation

Main objectives :

- Allow anyone who owns an **Arduino Vidor** board to have a **logic analyzer** for a investment less than CHF 100
- Demonstrate that it is possible to obtain a result **without knowledge of the Verilog language**
- Provide a **skeleton** for the Microsoft **WPF** (Windows Presentation Foundation) with ***dynamic DrawLine, PreviewKeyDown & Serial("COM", 800000)***
- Present some effective methods to **debug** such a project
- Replace my LabNation analyzer with a boring Trigger mechanism
- Provide **https://github.com/gelit/Logic-Analyzer-with-Arduino-Vidor** with all files

Why a logic analyzer ?

The logic analyzer is the essential tool for **testing and validating hardware developments**
It allows, **in a time window**, for example of 1670 µs (= micro-second), to visualize several digital signals :



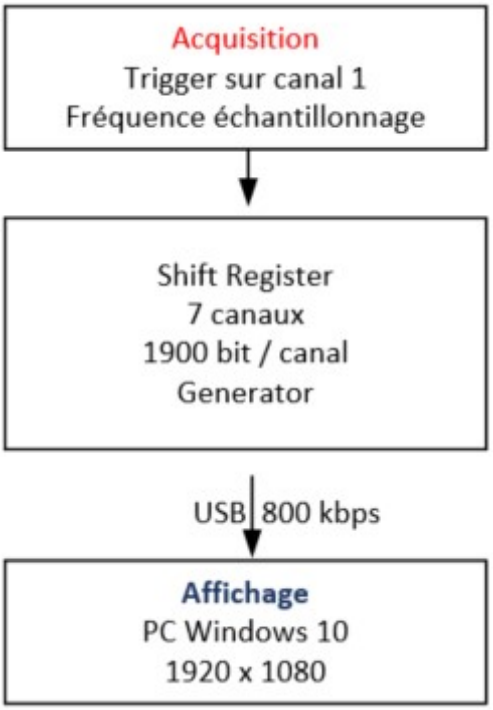
This example visualizes 7 channels (= digital signals)

The logic analyzer only knows states 0 and 1; there is therefore no need to attempt to measure a rise time
Unlike the vast majority of analysers, the proposed display does not display any edges !

Time = 396 indicates the time interval between the 2 cursors; the unit (µs or ms) is specified at the bottom with 1670 µs

Pos gives the pixel value of the 1st cursor (useful to check that the horizontal resolution is equal to 1920 pixels)

Principe of operation



During the **acquisition** phase, the 7 signals are read (sampled) every 0.879 µs to fill a 1900 bit memory.
The acquisition window is therefore = 1900 x 0.879 = 1670 **µs (microsecond)** = 1.67 **ms (millisecond)**

The sampling frequency is, in this case, equal to 1.14 MHz since F=1/T (Frequency = inverse of the Period)

Acquisition is started (Trigger) on the first rising (positive) edge or falling (negative) edge of signal (channel) 1

At the **display** level :

- The chosen horizontal resolution (1920 pixel) associates one pixel for each 0.879 µs (sample)
- The vertical resolution displays 7 signals (channels)

Thanks to USB, it is conceivable to connect N cards to have a device displaying N x 6 inputs + common channel 1 (Trigger)

Simple & effective use

Failure to detect the card is signaled by a beep and an explicit message NO DETECTION !
instead of Vigor detected

The first column displays the status of each input with a 1 second refresh

It is therefore useless to hope for an acquisition if channel 1 remains at L (Low) or H (High)

The default display places the Trigger edge at pixel 200; i.e. about 10% of the 1900 available
Keyboard shortcuts :

- **B** (Begin)
- **C** (Center) with edge at 50%
- **E** (End) 90%

S (Stop) locks the acquisition in order to keep the data of the last acquisition

State indicated by Stopped or Running (default mode)

The **acquisition window** is voluntarily limited to the width of the screen (1900 pixels) and varies according to the sampling frequency chosen with the **up and down arrows** :

- Min = 104 µs
- Max = 107 ms

La position du curseur 1 est affichée en première ligne
Horizontal arrows move cursors
Tab key selects cursor 1 or 2
The LeftShift key accelerates movement by 10
Cursor position 1 is displayed on the first line
The time interval, between the 2 cursors, is displayed
Type Z to hide cursors

The software integrates a signal generator with output on A0 whose sequences are accessible with keys 1 to 9

It offers various (easily editable) choices in the Arduino Sketch

```
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 : the card has 22 pins which are all used

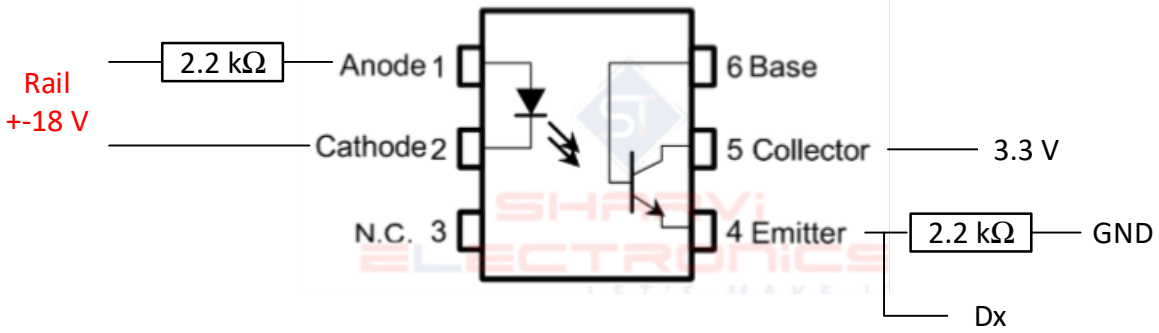
	Available	Interne use
• D0		Tshift (FPGA → CPU)
• D1-D7	7 input	
• D8-D14		7 output (FPGA → CPU)
• A0	Generator	
• A1		Byte Available (FPGA → CPU)
• A2-A4		Féch Sampling Frequency (FPGA ← CPU)
• A5-A6		Mode du Trigger (FPGA ← CPU)

Attention – Danger !!!
The person using this logic analyzer must understand the characteristics (limites) of a Dx input
According to
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

The max voltage is limited to 3.6 V

During my measurements on the Märklin rail, with an alternating voltage of +18V- with -18V+,
I use a CNY17 optocoupler in order to limit the voltage to 3.3 V and ensure galvanic separation



Choices

They facilitate the use of this logic analyzer :

- Only one Trigger possible with channel 1 on the rising or falling edge
- All the memory of the 7 channels of 1900 bit is displayed
- **Acquisition windows**

µs	ms	Tech ns	Tech ms	Féch MHz	Féch 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]	

Measurements performed with LabNation precision

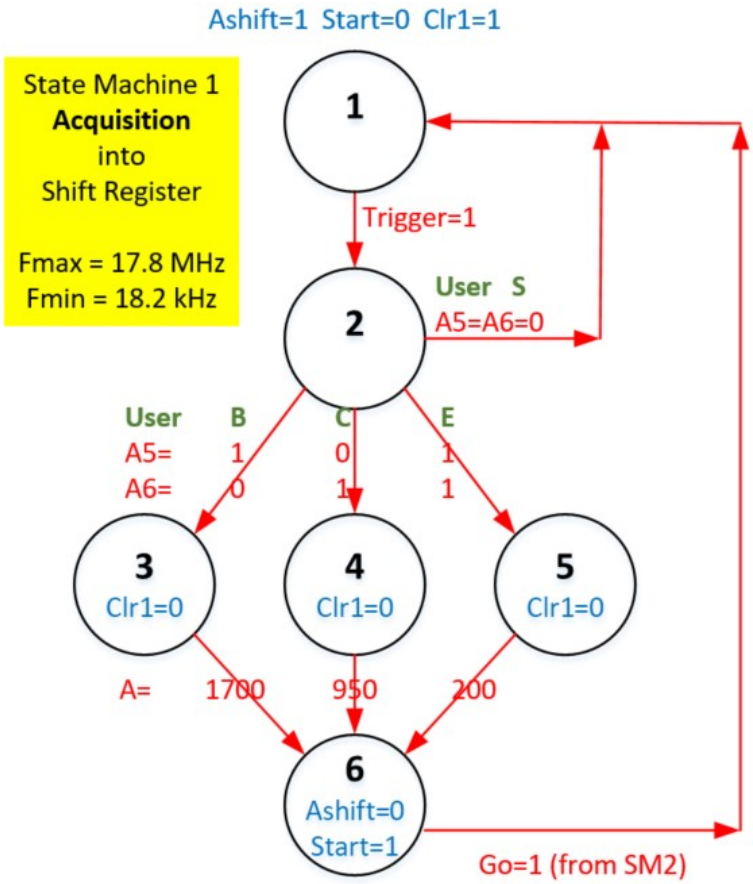
- 2 cursors to measure a time interval
- Trigger in Begin – Center – End or Stop mode
- Signal generator (output A0) easily modifiable
- Keyboarf shortcuts → no mouse

Changing the sampling rate

It is essential to understand that this change will result in a new display !
With the down arrow, the acquisition window increases (doubles) so that the display ends halfway.
On the other hand with the up arrow, the 1900 pixels of the new display correspond to half of the acquired data
This mechanism is iterative as long as the analyzer does not receive a new acquisition

It is therefore advisable, after modifying the acquisition window, to carry out a new acquisition in order to display the 1900 pixels of each channel

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



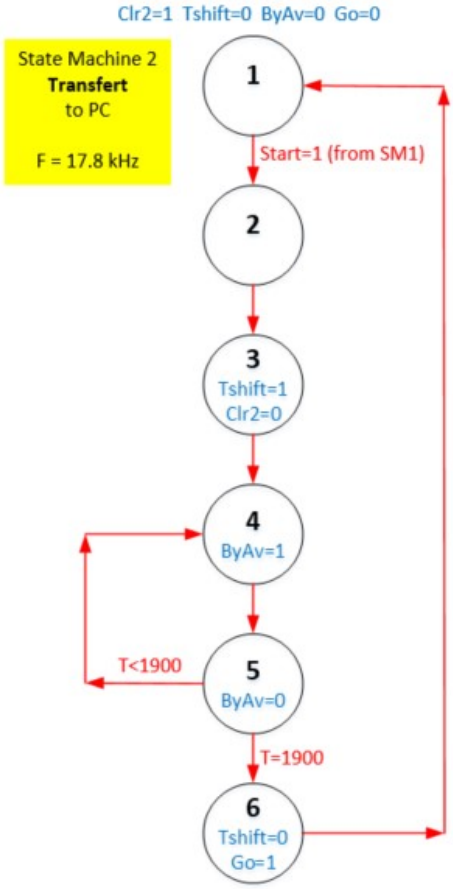
The **1st state machine (SM1)** manages the acquisition of data from the 7 channels in the shift registers.

In the initial state 1, Ashift=1 authorizes the acquisition
It changes to state 2 on the first positive or negative edge
→ Trigger=1

States 3 – 4 – 5 are chosen according to the mode Begin – Center – End selected

The default Begin mode displays this first edge of channel 1 at position (pixel) 200; which requires to acquire 1700 bit after the detection of the Trigger

Ashift=0 indicates the end of the acquisition
Start=1 starts the 2nd state machine
SM1 waits on Go=1 (generated by SM2)



The **2nd state machine (SM2)** manages the transfer of data to the PC via USB

It starts with Start=1 generated by SM1

In state 3, Tschift is activated to authorize the transfer (shift)

State 4 with ByAv (Byte Available) signals to the CPU that the data is available

In this synchronous mechanism, CPU must be able **in 56 ms (F=17.8 kHz)** to read the 7 channels and send the compressed data to the PC

State 6 terminates the transfer and places SM1 in its initial state

Format of data on USB

Visible in View – Output of VisualStudio thanks to `Debug.Write(Rx);`

Delta=3 Very useful during Debug to check the correct alignment → no lost bits
Just increase F to understand the problems

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

The display of the first page includes 7 channels of 1900 bit = 13'300 bit

Notepad++ indicates total length = 220 characters ; i.e. 2200 bit on USB (10 bit / caract.)

Le compression factor = 1 - 2200/13300 = 83 %

My tips for debugging :

ToggleLed allows to test if a procedure is called, the result of a test or … swithout impacting the CPU load

Serial1.print uses pins 13 – 14 and allows redirecting USB output to a putty terminal

Display states of State Machine by assigning outputs S0 – S1 – --- with powers of 2

Use this logic analyzer

Improvement points :

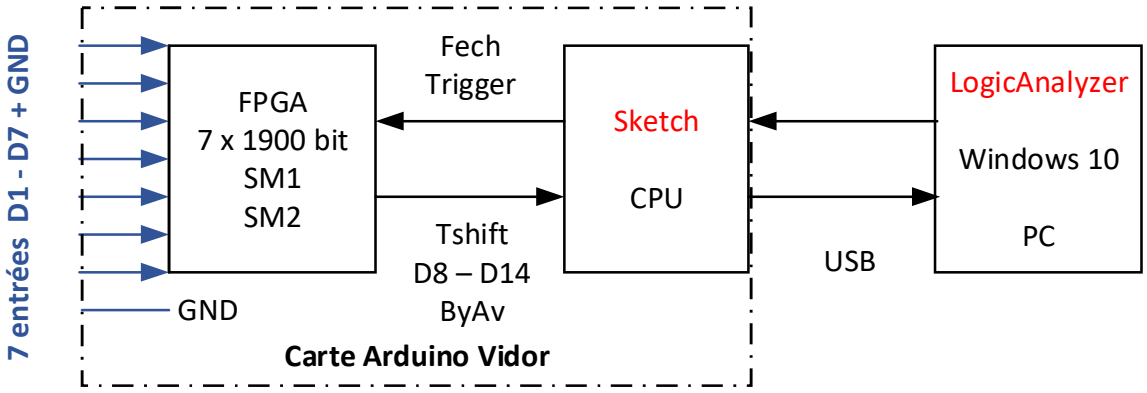
This project is intended to be **educational**; it therefore does not include all the functions of a high-end analyzer; **letting everyone add what they lack**

Some ideas that interest me :

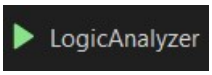
- Activate a sophisticated PLL *(PLL synthetizer)* to use sampling frequencies higher than 18 MHz ; forr exemple 350 MHz !
  Clock Tree Max = 402 MHz for 10CL016 … C8 according to
  https://www.intel.com/content/www/us/en/docs/programmable/683251/current/clock-tree-specifications.html
  PLL output frequency (C8) = 402.5 MHz according to
  https://www.intel.com/content/www/us/en/docs/programmable/683251/current/pll-specifications.html

- Is-it possible to read 8 digital input simultaneously ?
  My old Motorola 68000 allowed it very easy !!!

Overview

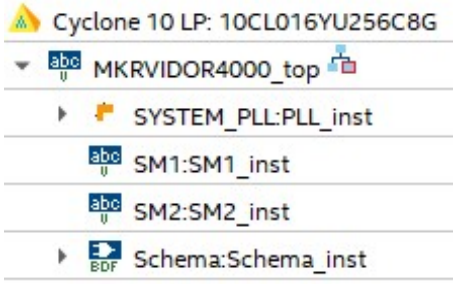


Minimalist implementation (without modification of the FPGA)

1. From https://github.com/gelit/Logic-Analyzer-with-Arduino-Vidor Clic Code – Download ZIP
2. Put uncompressed files into folder ...\Vidor\LA
3. Install & launch Arduino IDE 2.0
4. Identify the USB COMx port used
5. Load ...\Vidor\LA \Sketch\Sketch.ino
6. Install Visual Studio 2022
7. Project – Manage Nuget Packages tu update System.IO.Ports
8. View – Solution Explorer
9. Double-clic on MainWindow.xaml.cs
10. Ctrl F to search **COM**
11. Change the default value of 3 based on the result of 4
12. Compile with 

Implementation tu study or modify the FPGA

1. Have the basics of Intel Quartus software with https://gelit.ch/Vidor/Vidor1.pdf
2. Install & launch this software according page 5 of the document
3. Browse the 4 main files of this project : _top, SM1, SM2, Schema



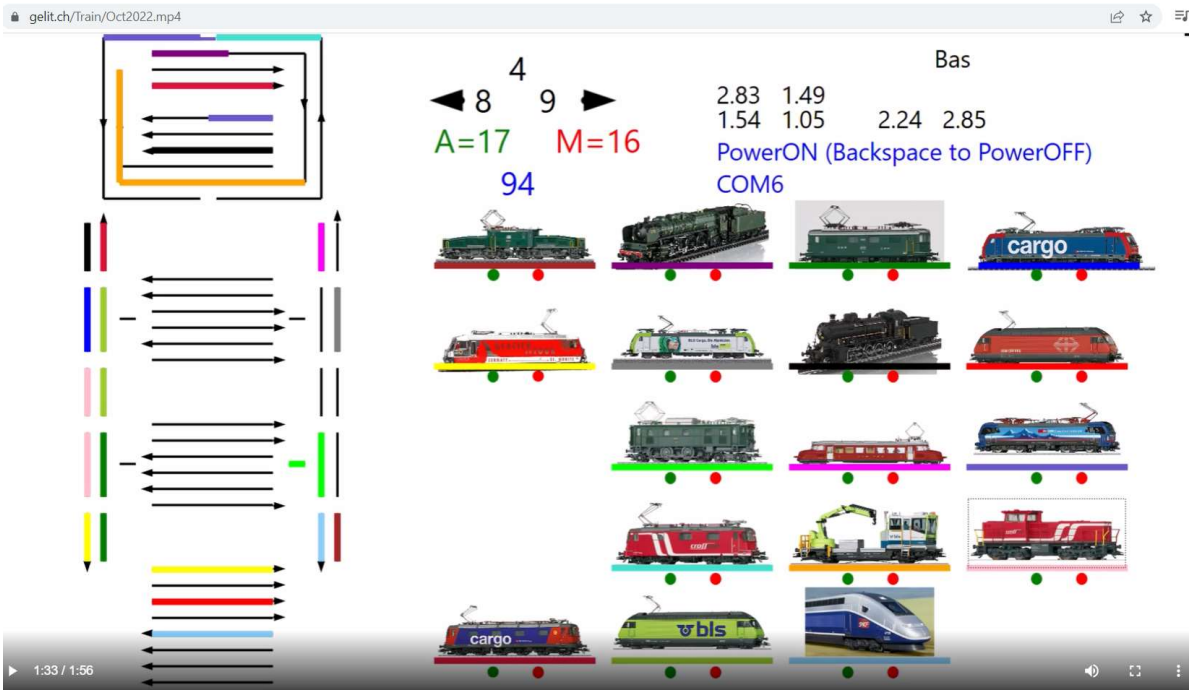
FPGA technology has revolutionized hardware development by doing away with stock ICs and wrapper gun
The excellent **free** Intel Quartus tool combined with libraries such https://flex.phys.tohoku.ac.jp/riron/vhdl/up1/altera/cat/lpm.pdf offers the person motivated by hardware development a beautiful playground.
Also, by owning 2 Vidor boards, this person will be able to develop on board 1 and test with the analyzer on board 2

Implementation tu study or modify the Windows programm (minimalist suite)

- In Solution Explorer, double-clic on MainWindow.xaml which contains the **static** definition of objects (Grid, Canvas, Text, Line) displayed
- MainWindow.xaml.cs contains the c# program

About MainWindow.xaml :

- The vast majority of functional examples found on the internet manage WPF (Windows Presentation Foundation) graphical objects in **static** ways !!!
- Luckily Google helped me to find this wonderful example https://www.youtube.com/watch?v=cvfkz0s6czA to **dynamically** write a line !!!
- My current implementation with cursor avoids overlapping it with the 7 displayed channels because I haven't found the zindex equivalent for static objects so far
- I planned for 2023 to use the HDMI compatible graphics library of the Vidor card https://www.youtube.com/watch?v=QSbFltEfQBs to test its performances by remplacing my static Windows 10 WPF display of my model trains



See more on :
https://gelit.ch/Train/Video.mp4
https://gelit.ch/Train/H4.pdf

In conclusion I am happy to announce, thanks to the valuable Intel Quartus Lite tool, that it is possible to develop an FPGA solution without knowledge of the Verilog language.

The only difficulty, easily manageable for a teenager, consists in correctly defining input, output and wire in the file https://github.com/gelit/Logic-Analyzer-with-Arduino-Vidor/blob/main/VidorFPGA-master/projects/MKRVIDOR4000_template/MKRVIDOR4000_top.v

Intel Quartus Lite continues the philosophy of great Arduino products by combining simplicity, quality and price affordability

Intel remains the hardware leader for me. This company, with a very high reputation, should fix some bugs that an average user observes after 1 hour of use.

I remain at disposal for any further information.

Gérald Litzistorf – retired professor – https://gelit.ch/