@EchelleInconnue
@VentresMous

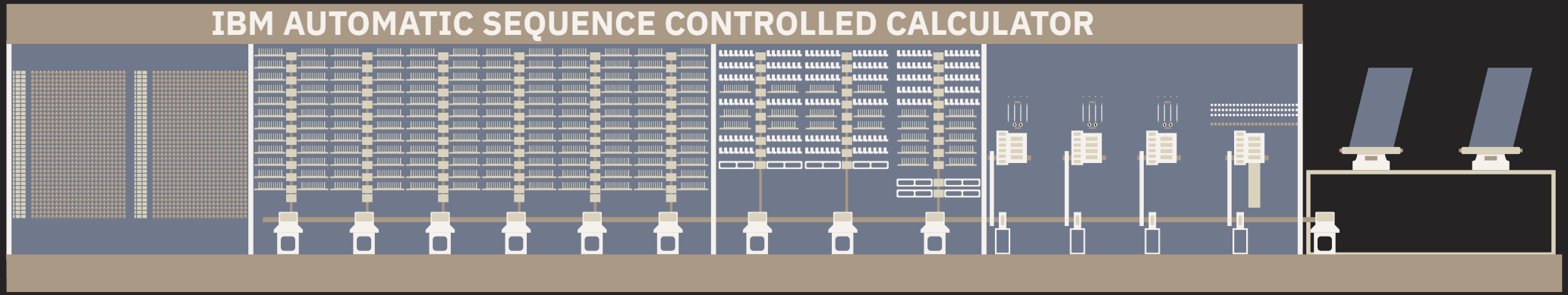
FPGA

Créer du matériel en programmant

SOMMAIRE

IL ÉTAIT UNE FOIS : LES CPU

15,5 mètres



constantes

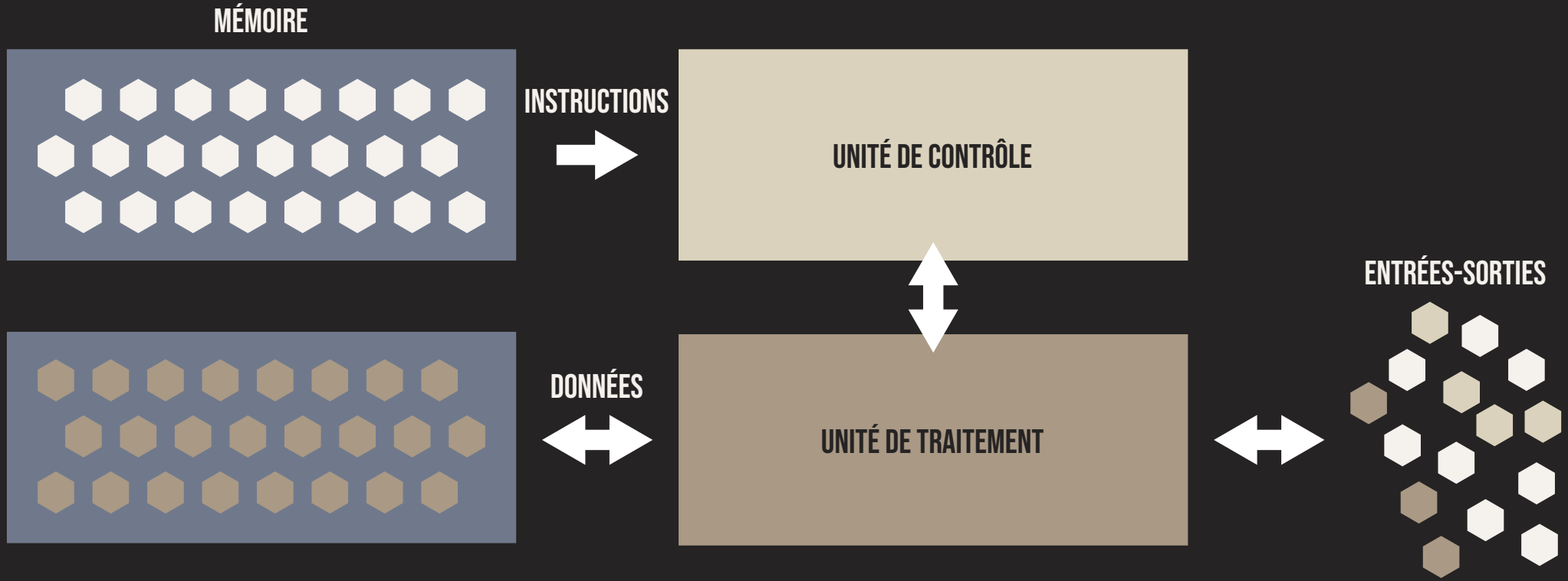
mémoire de données

unité de traitement

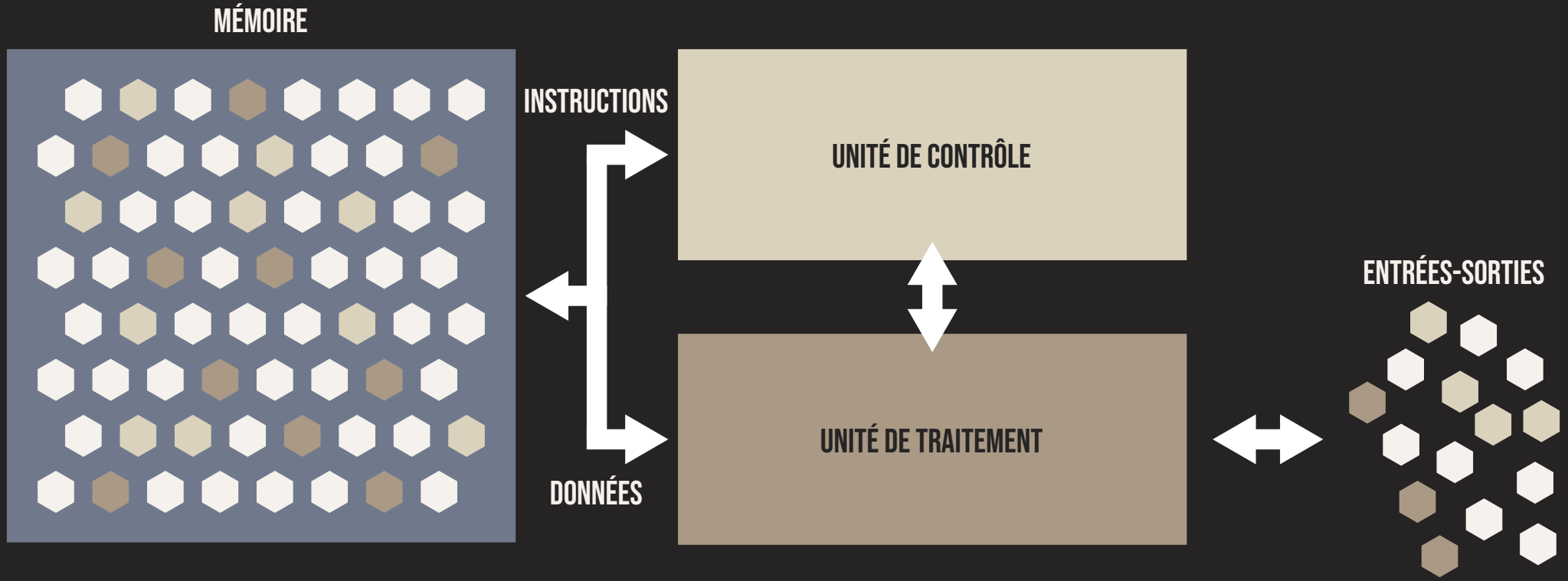
unité de contrôle
mémoire d'instructions

entrées/sorties

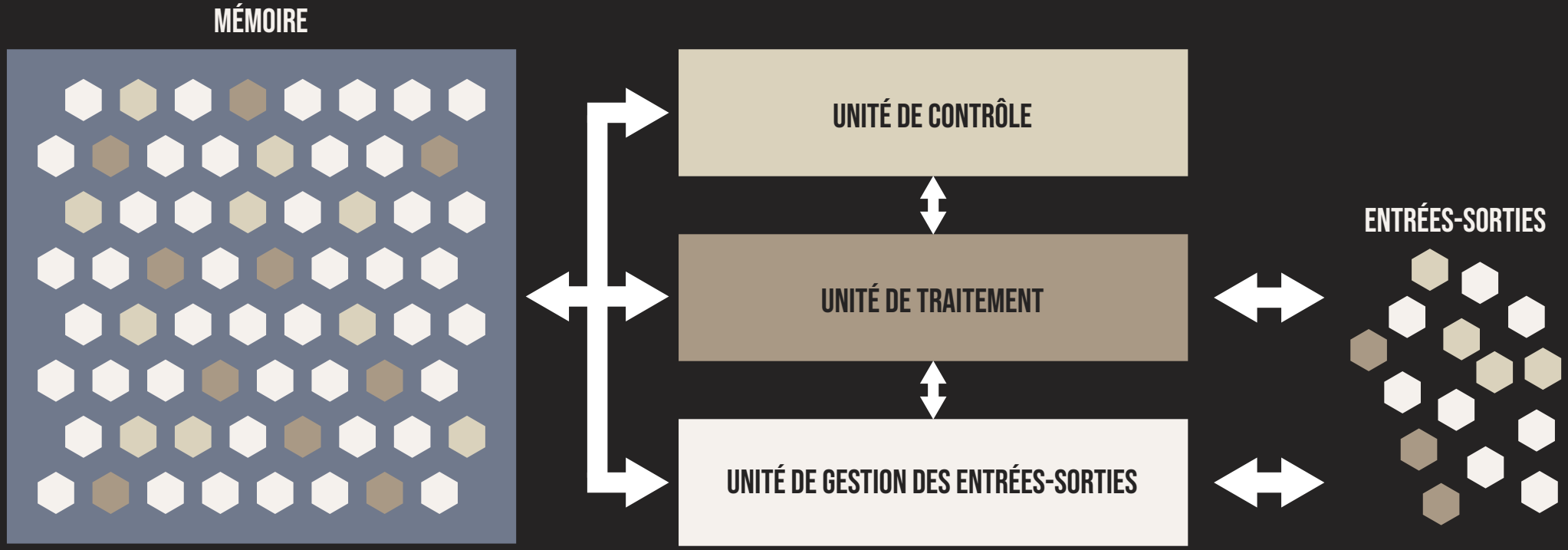
HARVARD MARK I, 1944



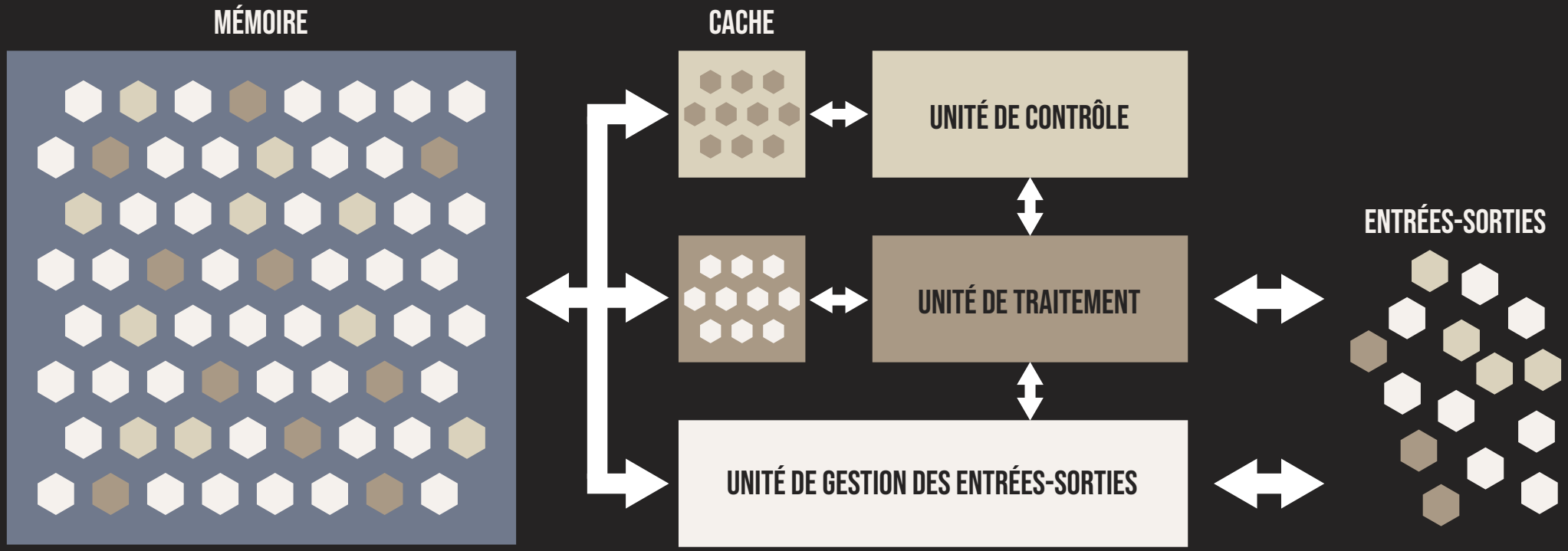
ARCHITECTURE HARVARD



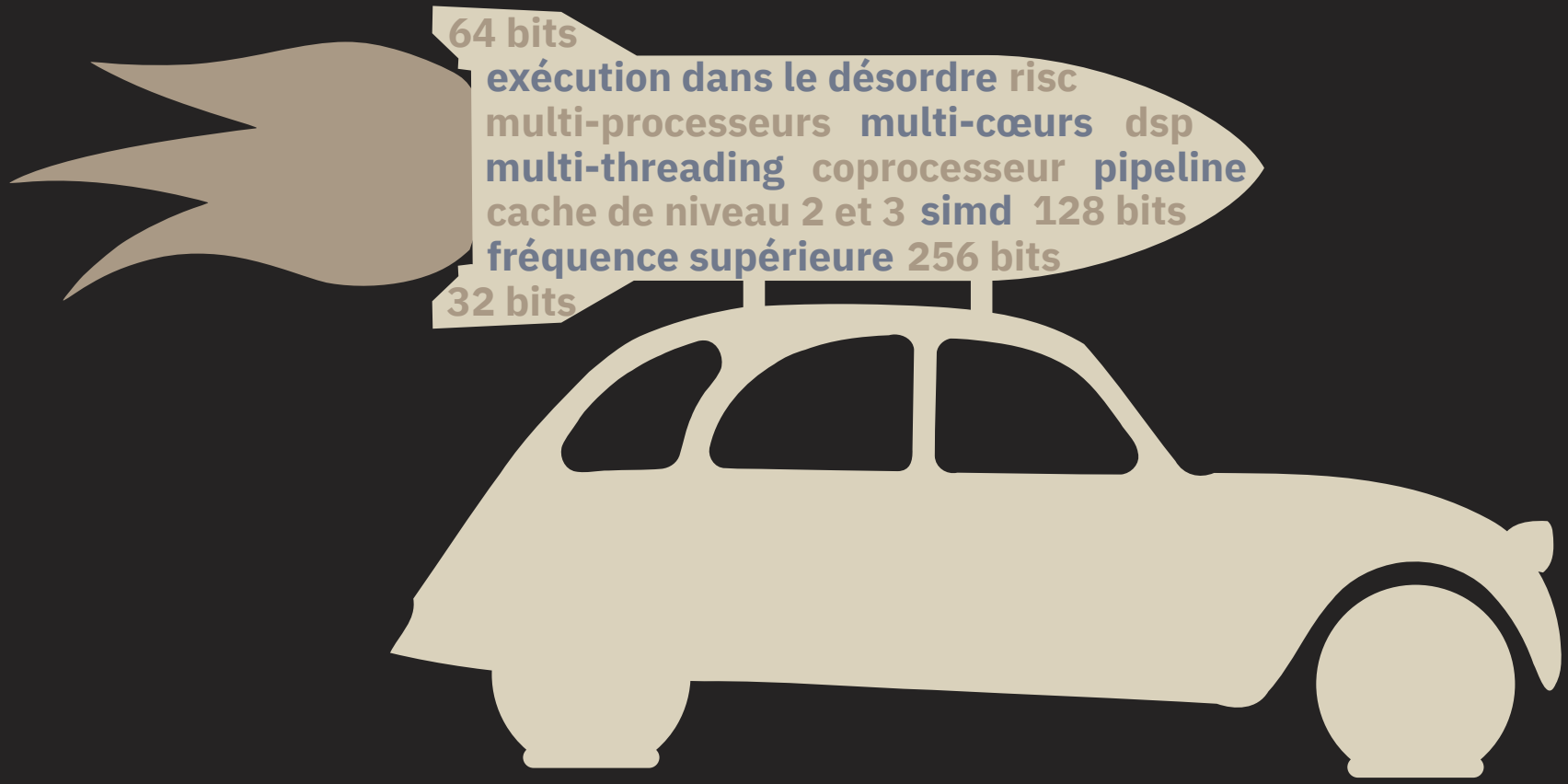
ARCHITECTURE DE VON NEUMANN



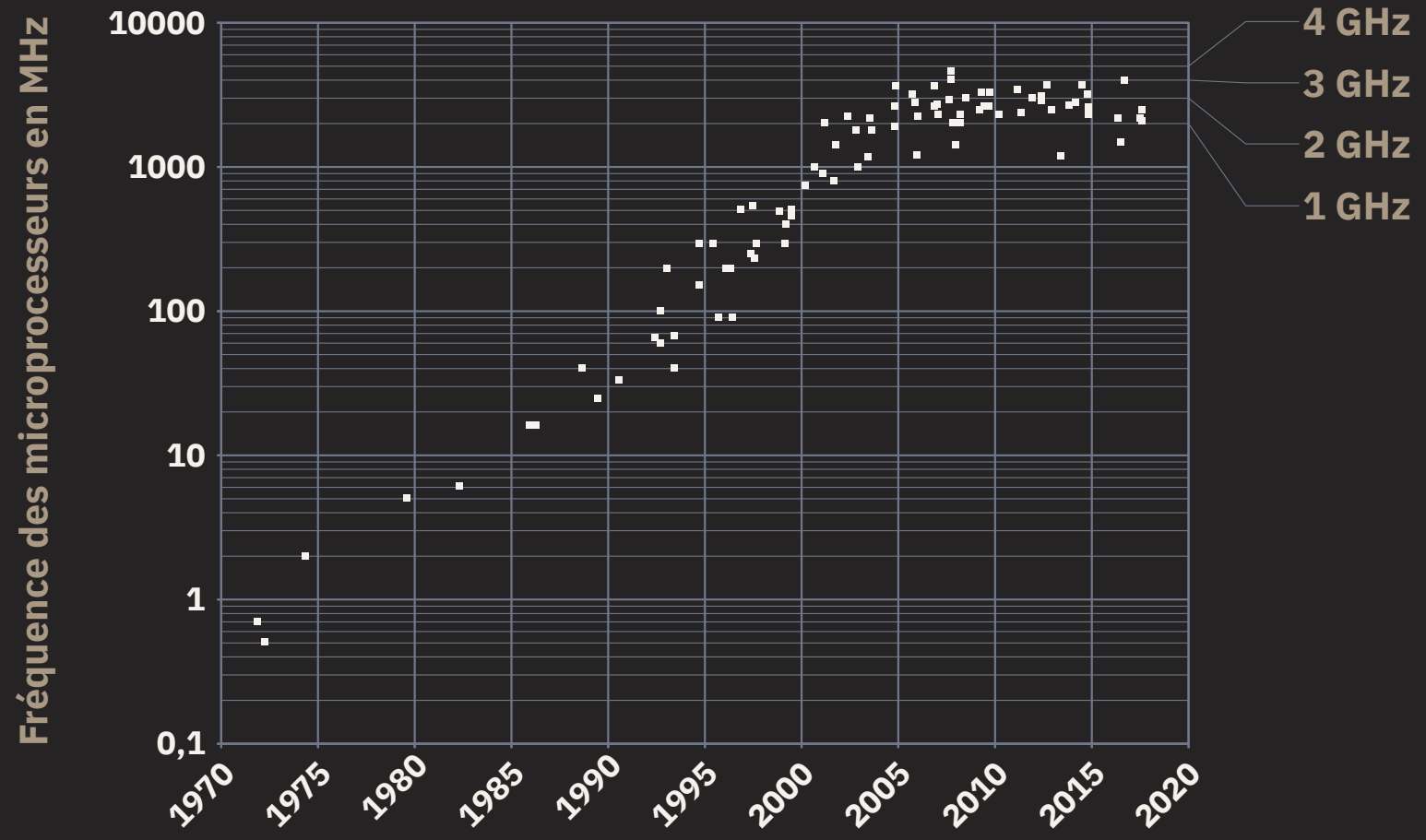
ÉVITONS LES PROCESSEURS PASSE-PLAT !



SOULAGEONS LE BUS DE DONNÉES

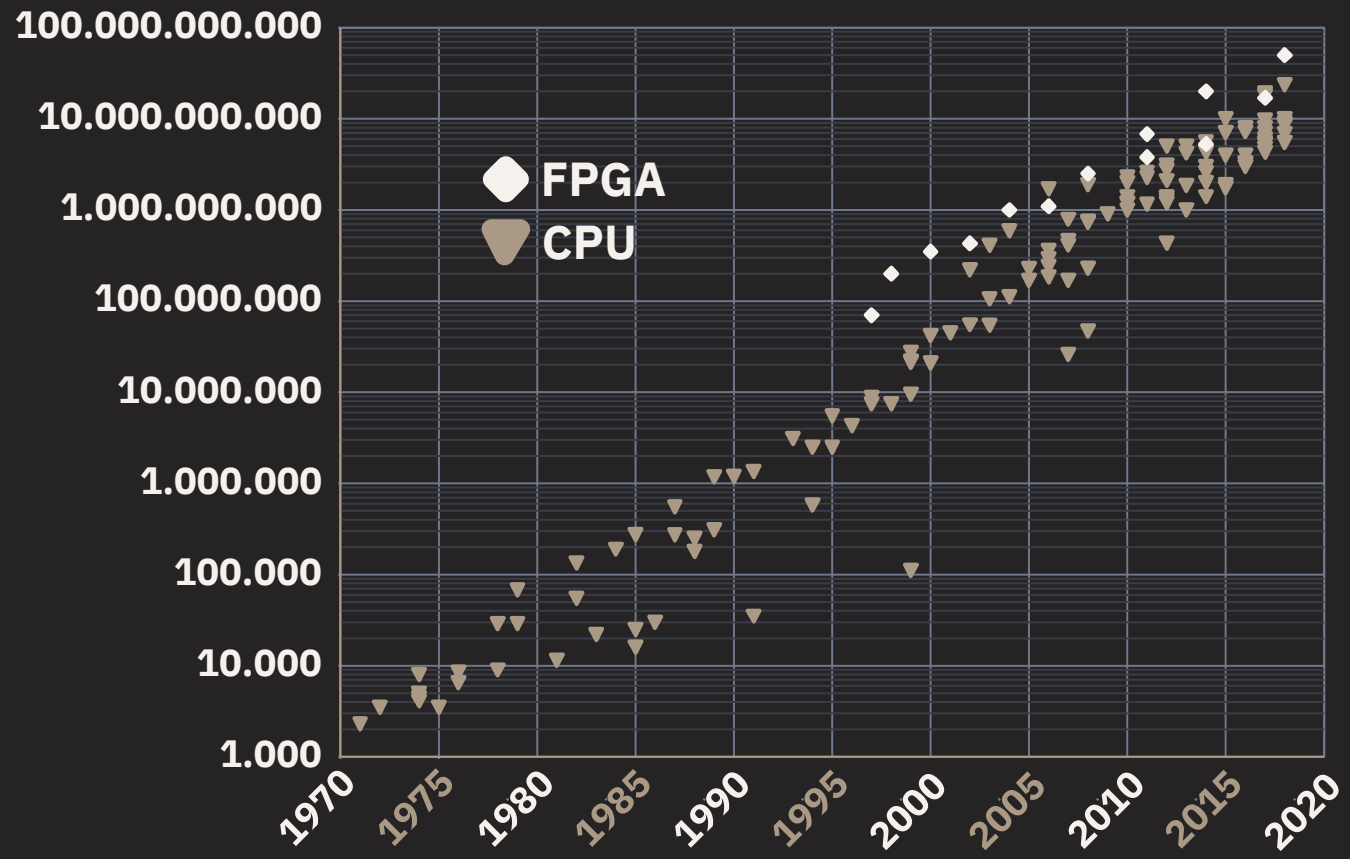


ACCÉLÉRER UNE 2CV N'EN FAIT PAS UNE FUSÉE



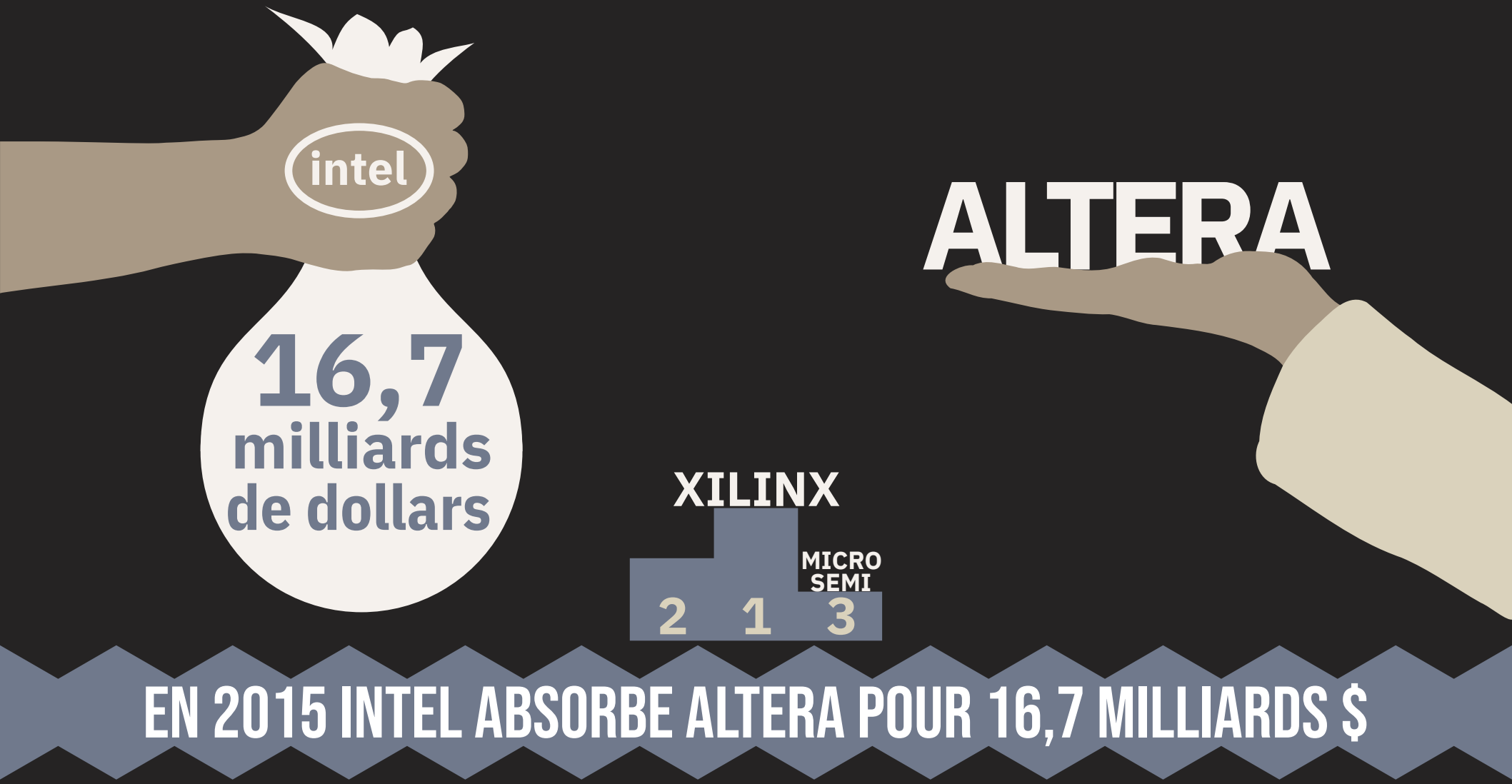
LES FRÉQUENCES STAGNENT DEPUIS 2005

Source: Karl Rupp, microprocessor trend data
Creative Commons Attribution 4.0 International Public License, 15/02/2018
https://github.com/karlrupp/microprocessor-trend-data

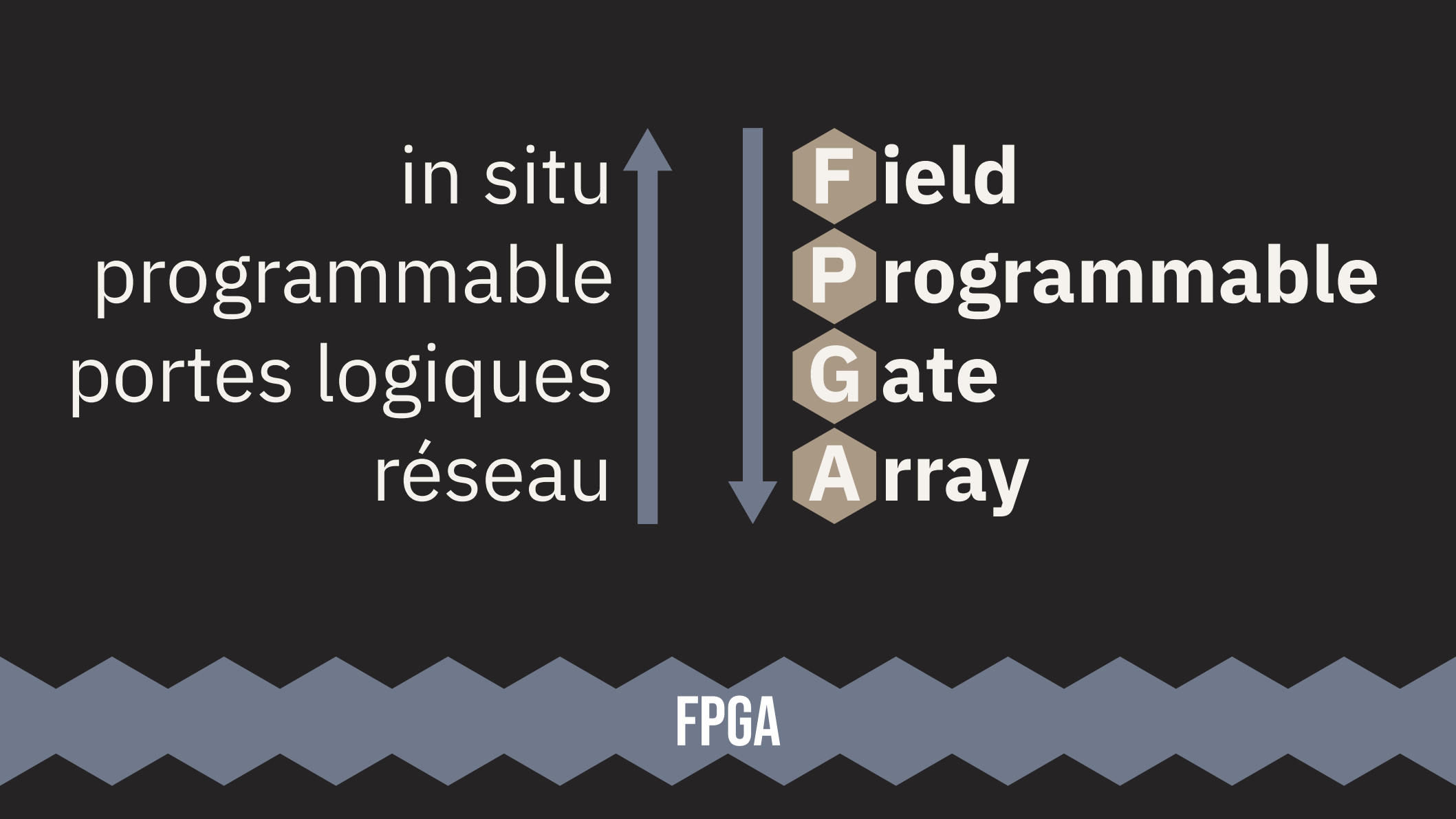


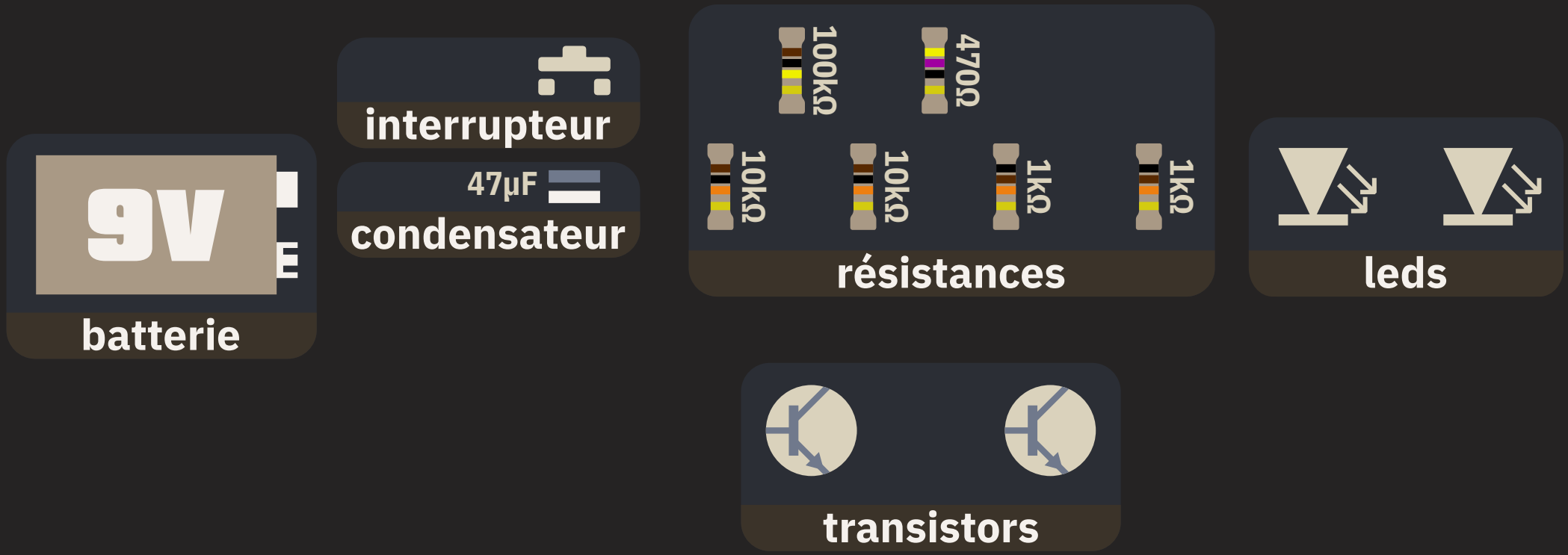
Source: Transistor count, 30/05/2019
https://en.wikipedia.org/wiki/Transistor_count

LA LOI DE MOORE A PERDURÉ JUSQU'EN 2017

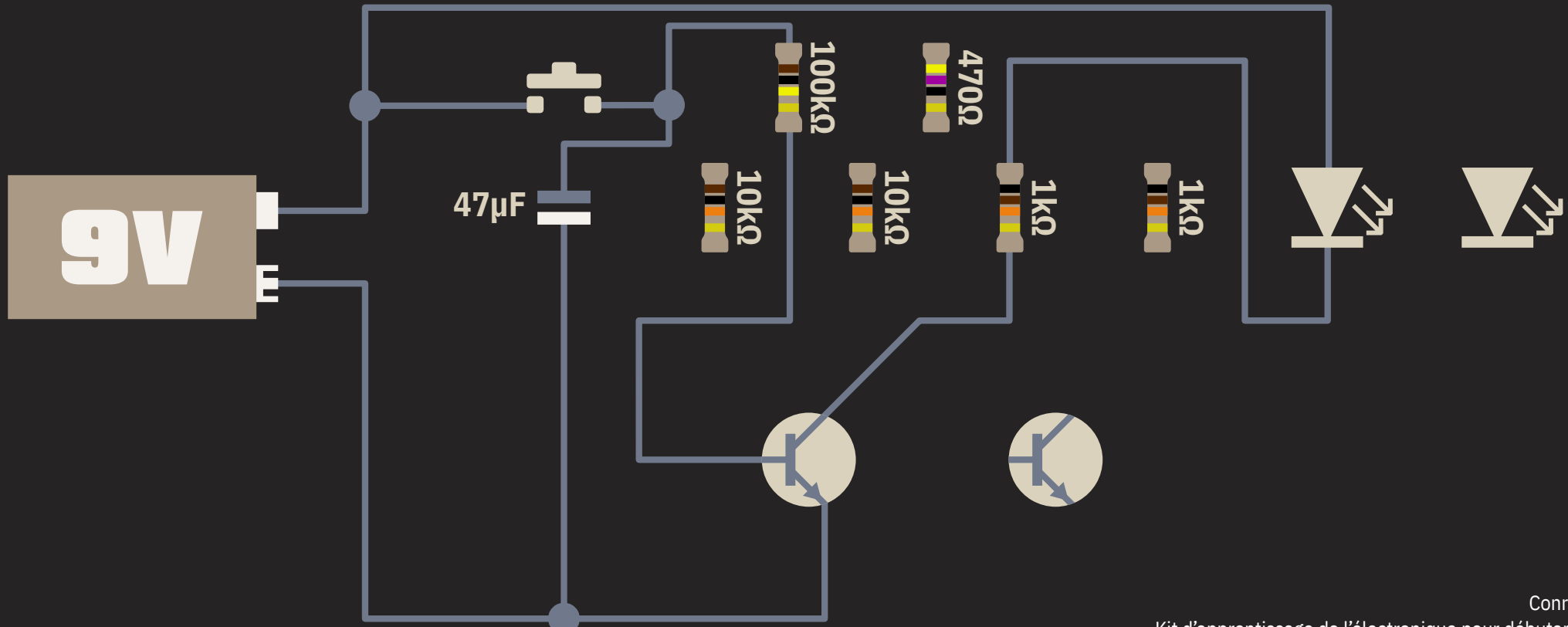

intel
16,7
milliards
de dollars
ALTERA
XILINX
MICRO
SEMI
2
1
3
EN 2015 INTEL ABSORBE ALTERA POUR 16,7 MILLIARDS $

C’EST QUOI UN FPGA ?


in situ
programmable
portes logiques
réseau
Field
Programmable
Gate
Array
FPGA

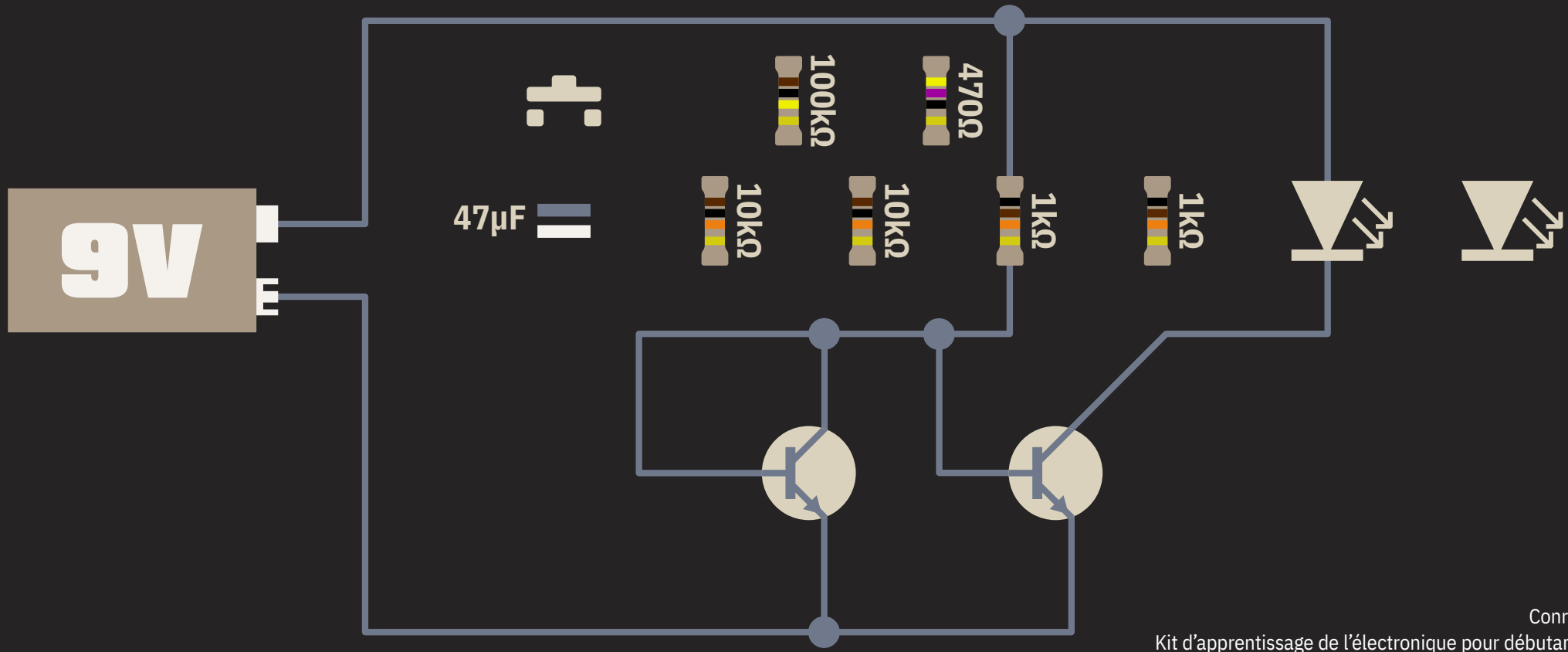


IMAGINEZ DES COMPOSANTS ÉLECTRONIQUES



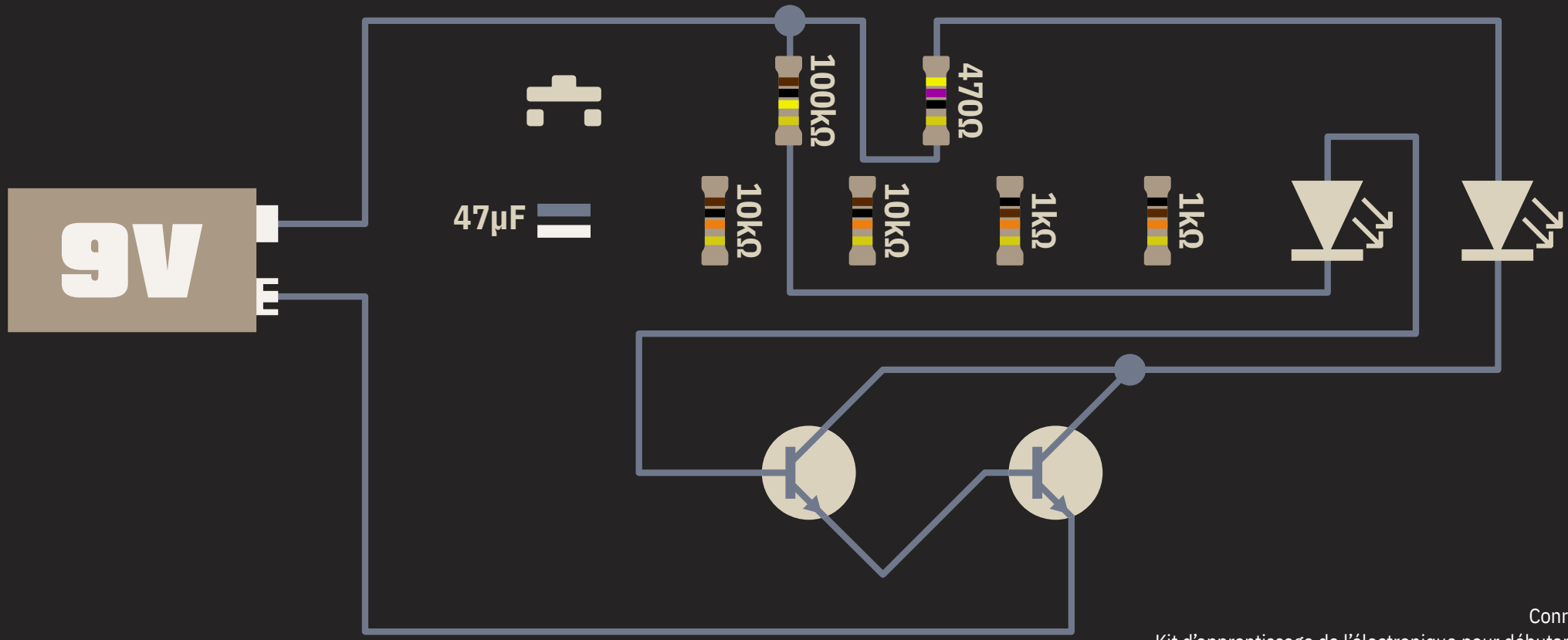
Conrad
Kit d'apprentissage de l'électronique pour débutants
https://produktinfo.conrad.com/datenblaetter/175000-199999/192230-an-01-fr-LERNPAKET_25_ELEKTRONIK_EXPERIMENTE.pdf

ILS PEUVENT FORMER UN RETARDATEUR



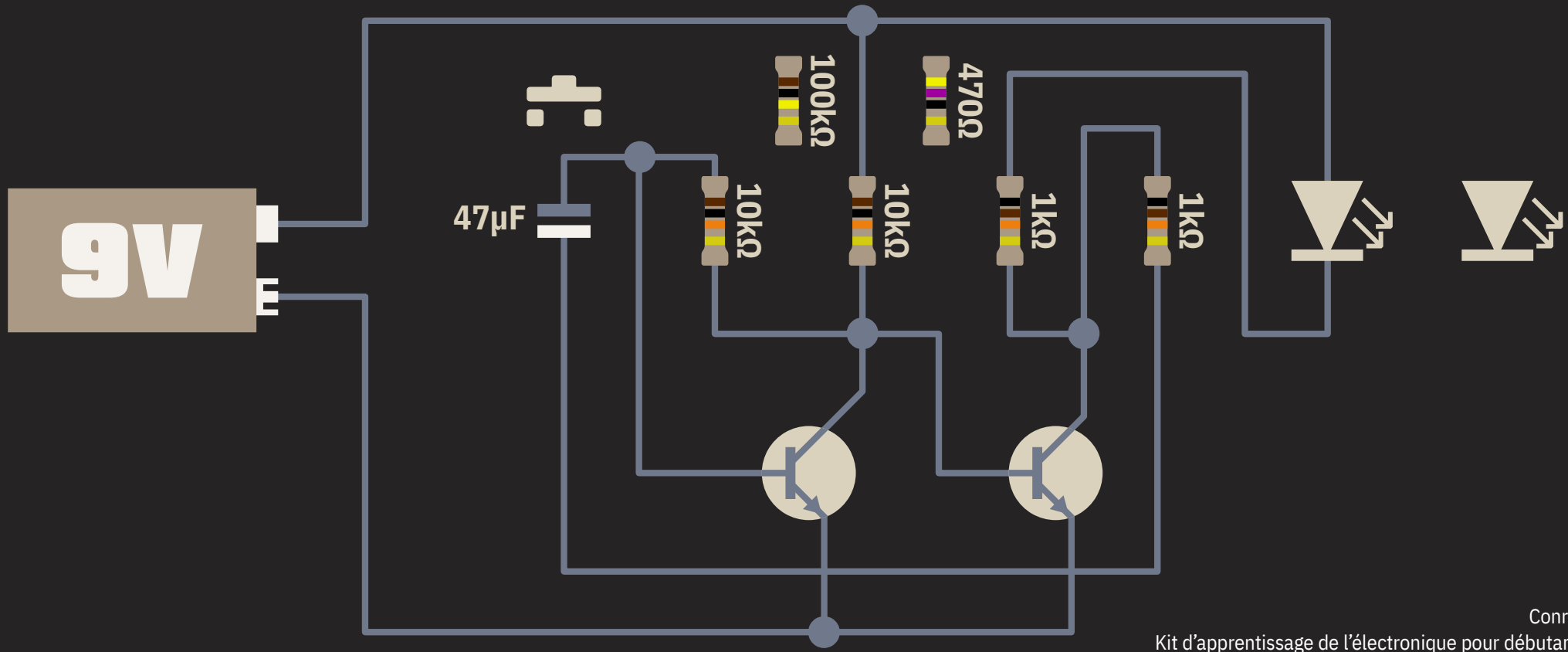
Conrad
Kit d'apprentissage de l'électronique pour débutants
https://produktinfo.conrad.com/datenblaetter/175000-199999/192230-an-01-fr-LERNPAKET_25_ELEKTRONIK_EXPERIMENTE.pdf

UN CAPTEUR DE TEMPÉRATURE



Conrad
Kit d'apprentissage de l'électronique pour débutants
https://produktinfo.conrad.com/datenblaetter/175000-199999/192230-an-01-fr-LERNPAKET_25_ELEKTRONIK_EXPERIMENTE.pdf

UN CAPTEUR DE LUMIÈRE



Conrad
Kit d'apprentissage de l'électronique pour débutants
https://produktinfo.conrad.com/datenblaetter/175000-199999/192230-an-01-fr-LERNPAKET_25_ELEKTRONIK_EXPERIMENTE.pdf

OU ENCORE UNE LED CLIGNOTANTE

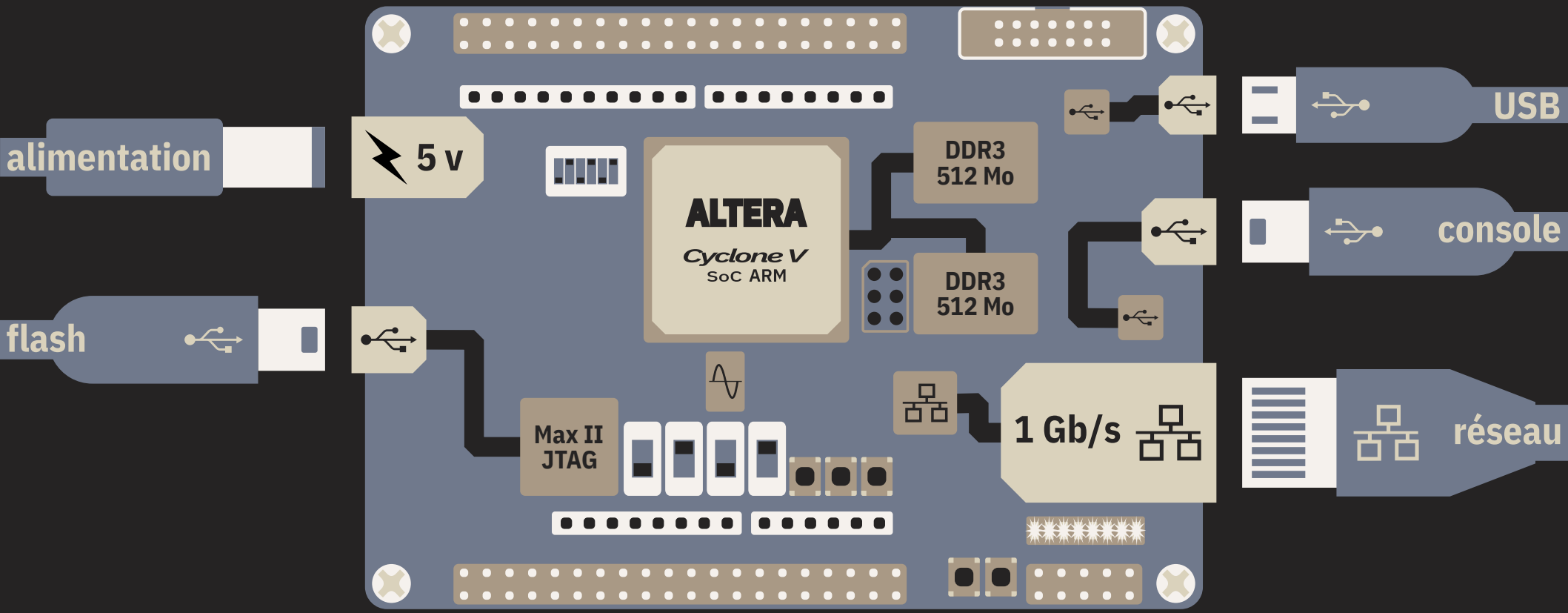
QUELQUES REMARQUES

- Le câblage définit le fonctionnement du circuit
- Les composants
  - restent fixes entre les différents schémas
  - ne sont pas tous utilisés pour un schéma donné
- C’est le principe d’un FPGA !

Réseau de portes logiques programmable in situ

- Un FPGA a 3 éléments constitutifs
  - des élément logiques (ALM/CLB, mémoire, DSP...)
  - un réseau de pistes
  - une mémoire de configuration
- Le réseau de pistes est figé
  - un FPGA reste un circuit intégré
  - toutes les combinaisons ne sont pas possibles
  - il est reconfigurable à volonté grâce à la mémoire

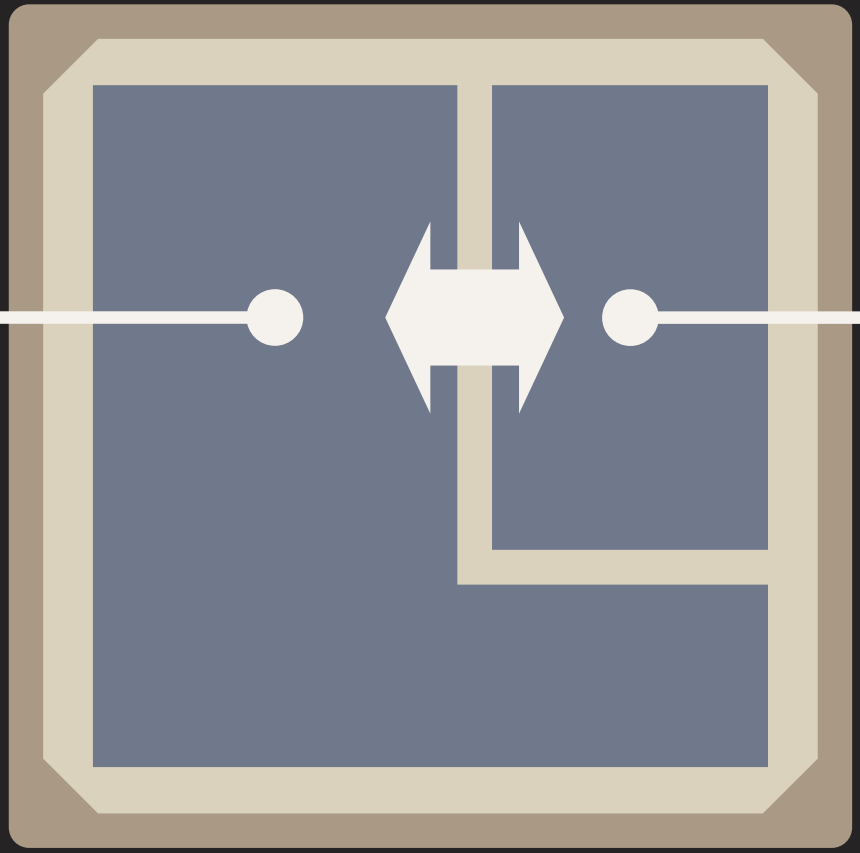
LE DEO-NANO-SOC



LA PLATEFORME D'INITIATION DE0-NANO-SOC

FPGA

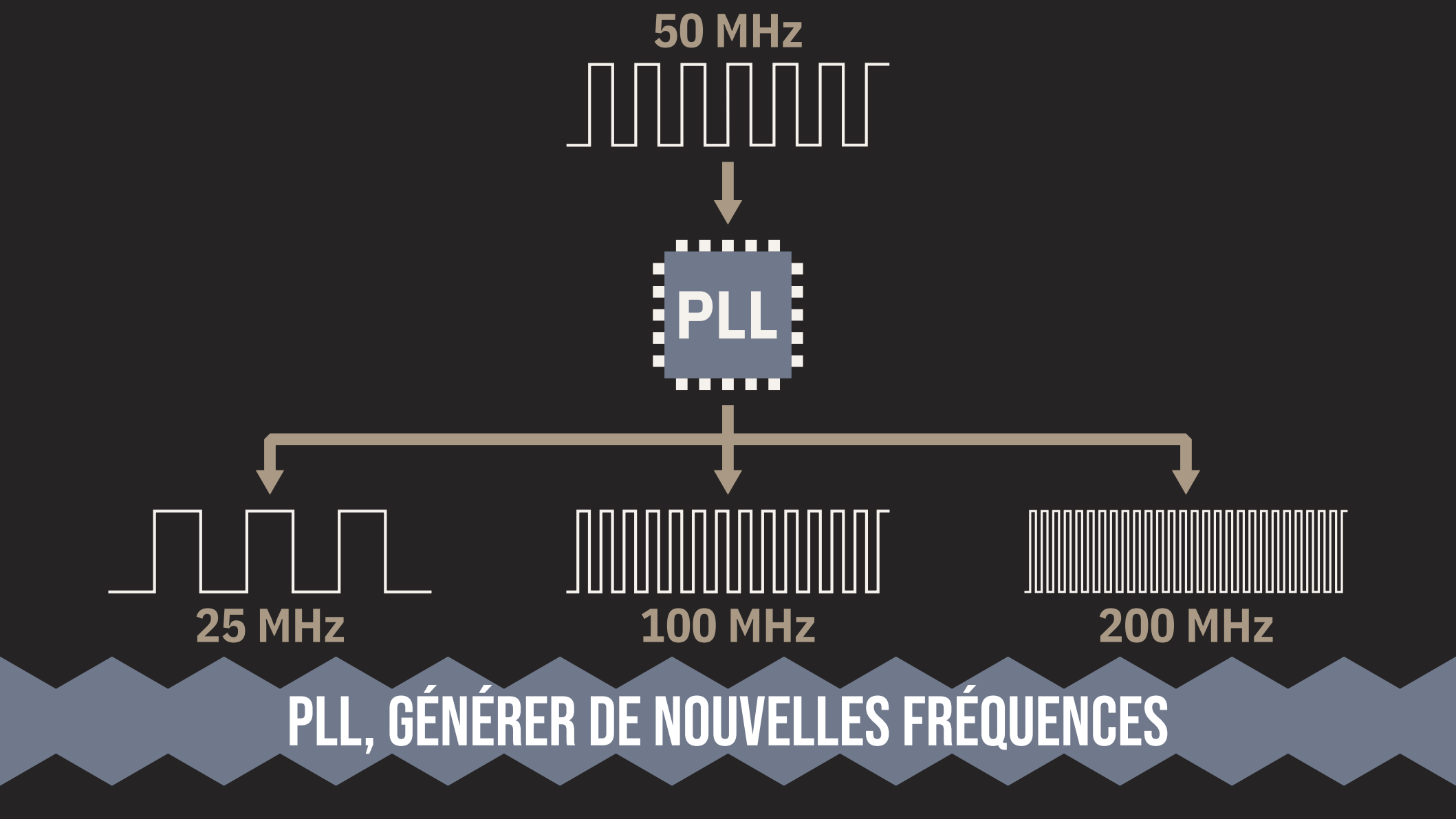
15880 CLB
330 Ko RAM
50 MHz



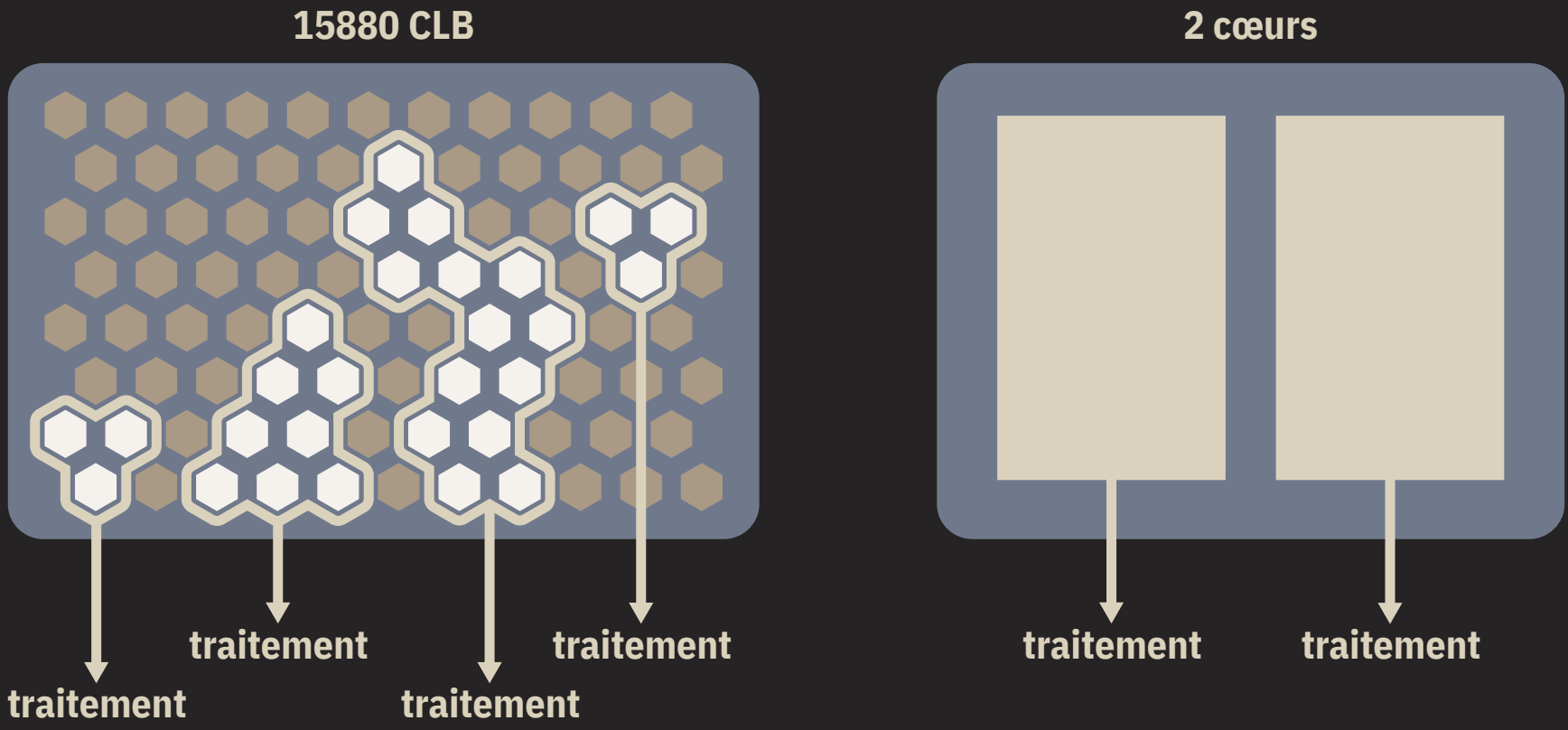
ARM Cortex-A9

2 cœurs
925 MHz

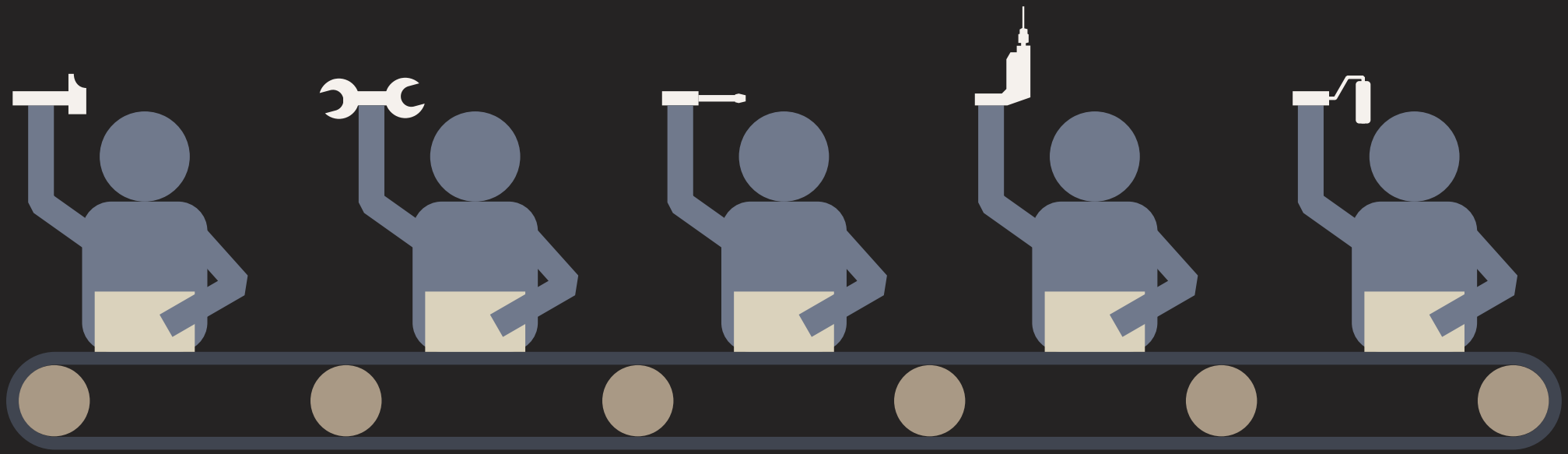
AU CŒUR DU DE0-NANO-SOC : LE CYCLONE V



PLL, GÉNÉRER DE NOUVELLES FRÉQUENCES



MODULAIRE ET EXTRÊMEMENT PARALLÈLE



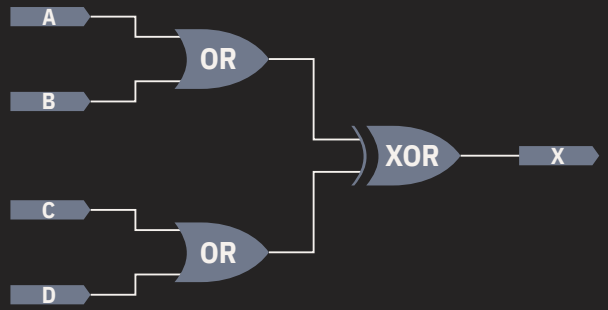
PIPELINE, TRAVAIL À LA CHAÎNE

COMMENT PROGRAMMER UN FPGA ?

LES OUTILS DISPONIBLES

- Chaque fabricant a ses propres outils
  - Xilinx → Vivado/ISE
  - Intel → Quartus Prime
  - Lattice → Diamond
  - etc.
- Ils sont **indispensables** pour générer l'image bitstream
  - ils sont gratuits pour les cartes d'initiation
  - les outils libres ne couvrent pas tout le flot de conception

dessiner un schéma



avec des portes logiques,
des composants électroniques

utiliser un HDL ou langage de description de matériel

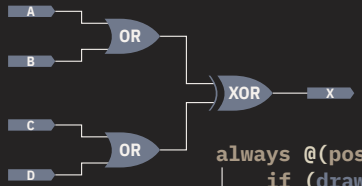
```
always @(posedge clk) begin
    if (draw) begin
        pixel <= 1'b1;
    end else begin
        pixel <= 1'b0;
    end
end
```

pour décrire le comportement
du circuit souhaité

utiliser OpenCL

pour utiliser un langage
de haut niveau

3 FAÇONS DE PROGRAMMER UN FPGA




```
always @(posedge clk) begin
    if (draw) begin
        pixel <= 1'b1;
    end else begin
        pixel <= 1'b0;
    end
end
```

simulation débogage

HDL

analyse synthèse

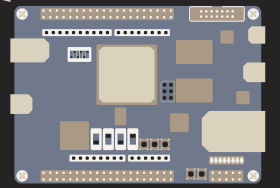
netlist

placement routage

assembleur

assemblage

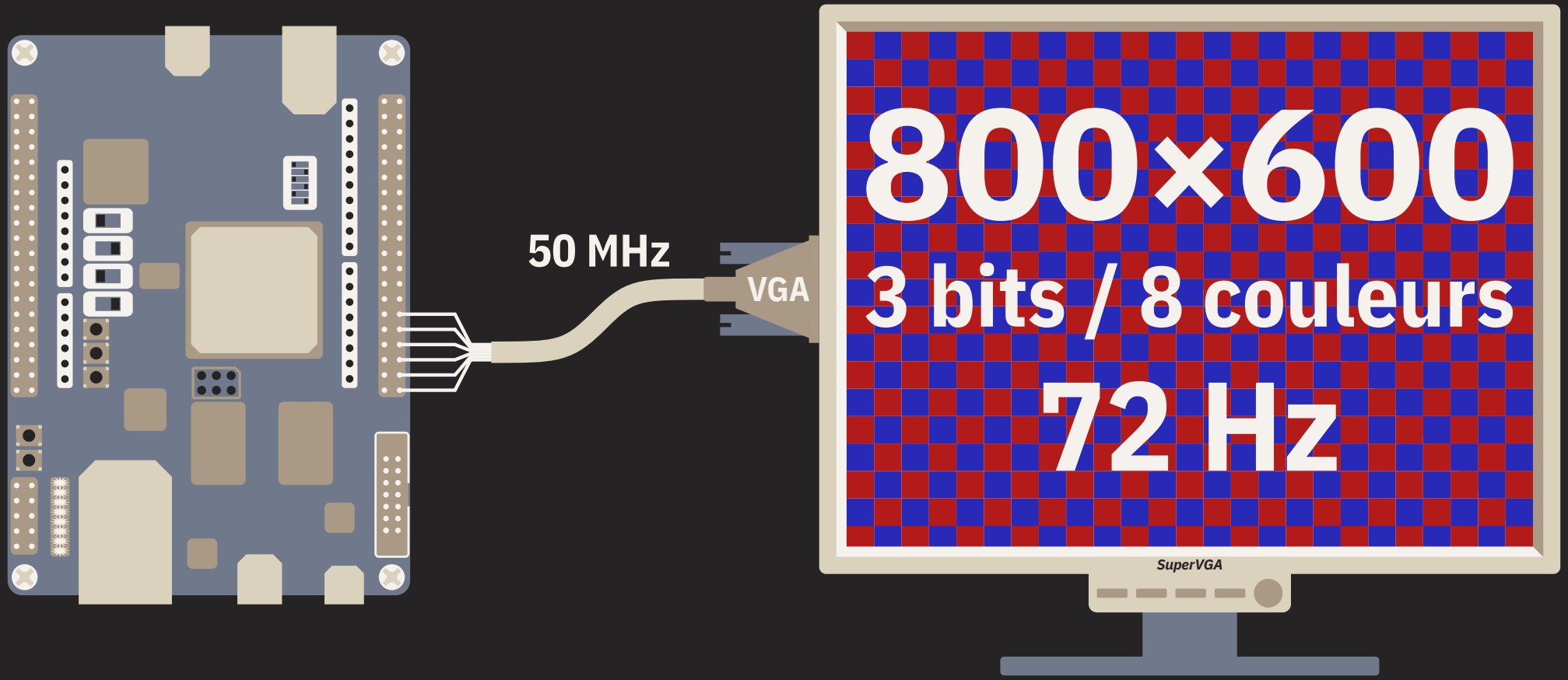
bitstream



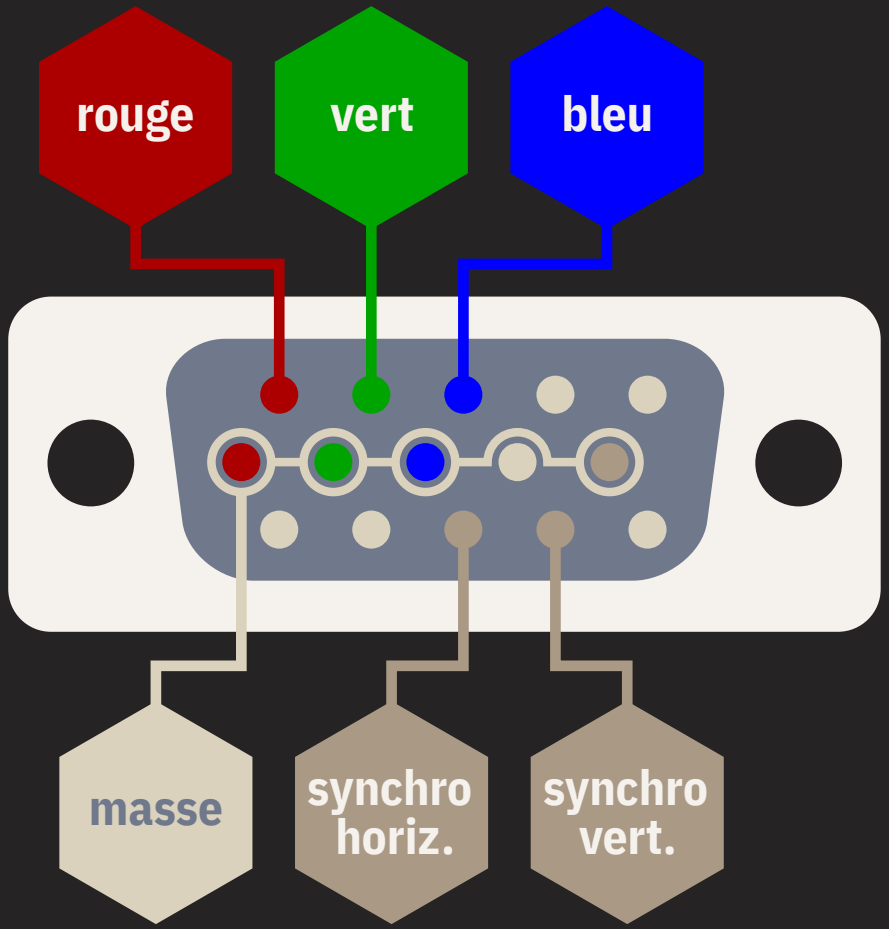
de quelques minutes à plusieurs heures !

LE FLOT DE CONCEPTION

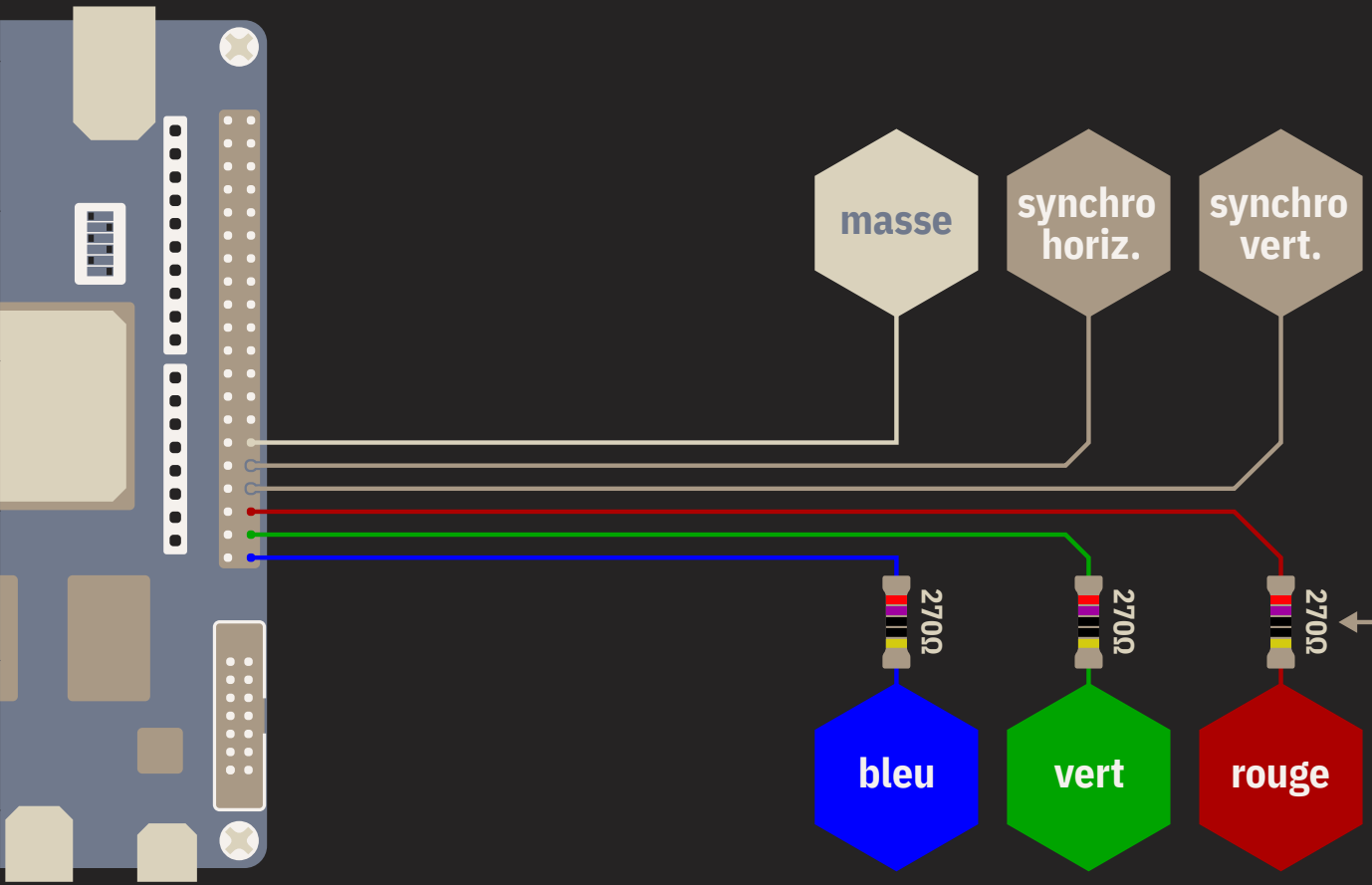
UN EXEMPLE : SIMPLEVGA



GÉNÉRATION D'UN SIGNAL VIDÉO VGA



LA PRISE VGA

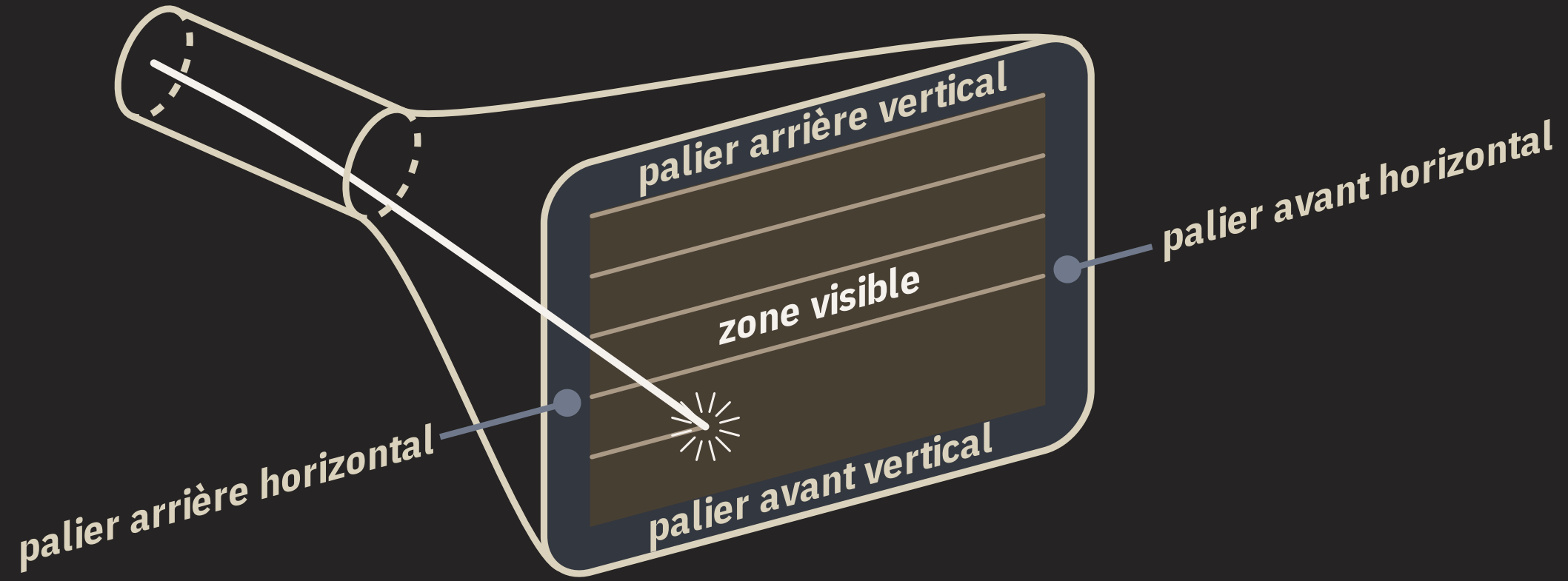


les broches du DE0-Nano-SoC utilisent une tension de 3,3 volts

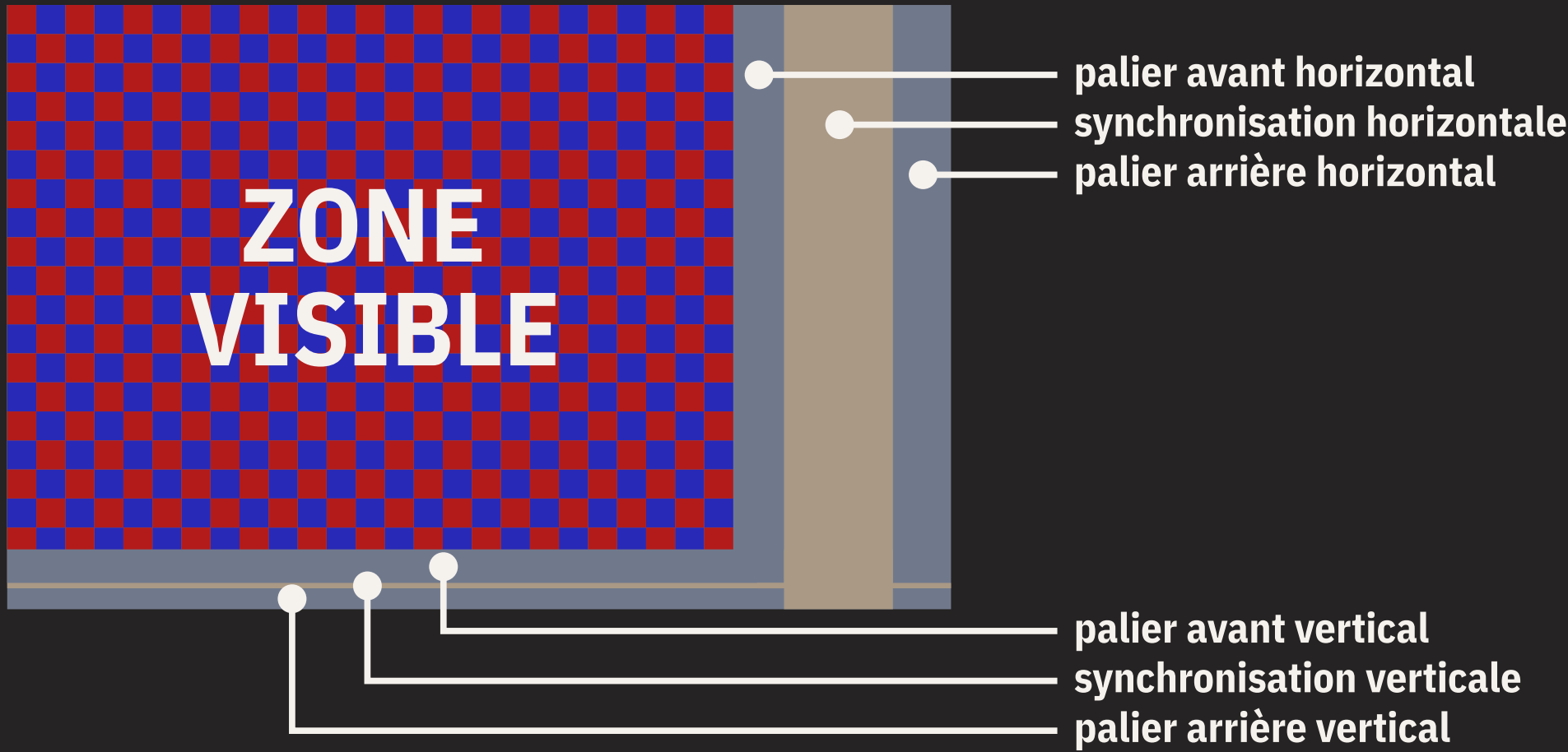
l'intensité lumineuse maximum est de 0,7 volts en VGA

les résistances permettent d'adapter la tension

CONNECTER LA PRISE VGA AU DE0-NANO-SOC



UN SIGNAL VIDÉO HÉRITÉ DES TUBES CATHODIQUES



LE SIGNAL VGA

ÉCRITURE DU CODE VERILOG

1) Déclaration du module
2) Synchronisation horizontale
3) Synchronisation verticale
4) Damier rouge et bleu

1

```
module SimpleVGA (
        input wire clk,

        output wire hsync,
        output wire vsync,
        output reg red,
        output reg green,
        output reg blue
);
```

2

```
reg [10:0] xpos = 0;
always @(posedge clk)
        if (xpos == 1039) xpos <= 0;
        else              xpos <= xpos + 1;

assign hsync = xpos < 856 || xpos >= 976;
```

3

```
reg [9:0] ypos = 0;
always @(posedge clk)
        if (xpos == 1039) begin
                if (ypos == 665) ypos <= 0;
                else             ypos <= ypos + 1;
        end

assign vsync = ypos < 637 || ypos >= 643;
```

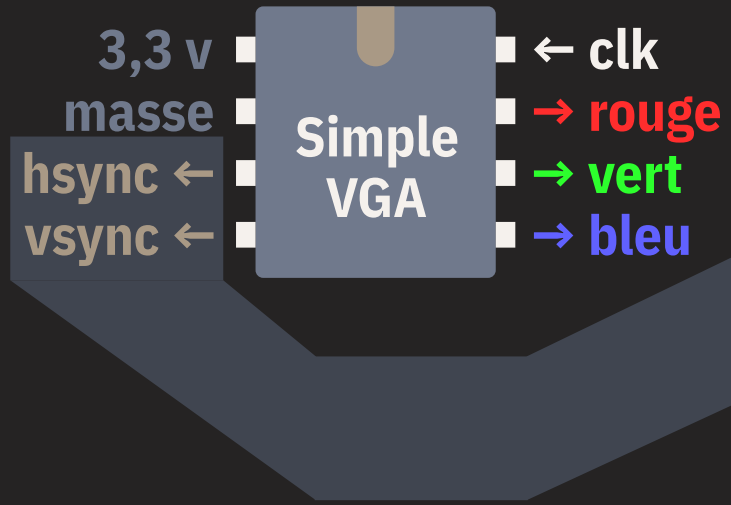
4

```
always @(posedge clk)
        if (xpos < 800 && ypos < 600) begin
                red   <= ~xpos[5] ^ ypos[5];
                green <= 0
                blue  <=  xpos[5] ^ ypos[5];
        end else begin
                red   <= 0;
                green <= 0;
                blue  <= 0;
        end

endmodule
```

DÉCLARATION DU MODULE


```
module SimpleVGA (
    input  wire clk,

    output wire hsync,
    output wire vsync,
    output reg  red,
    output reg  green,
    output reg  blue
);
```

CRÉATION DU CIRCUIT INTÉGRÉ, HORLOGE



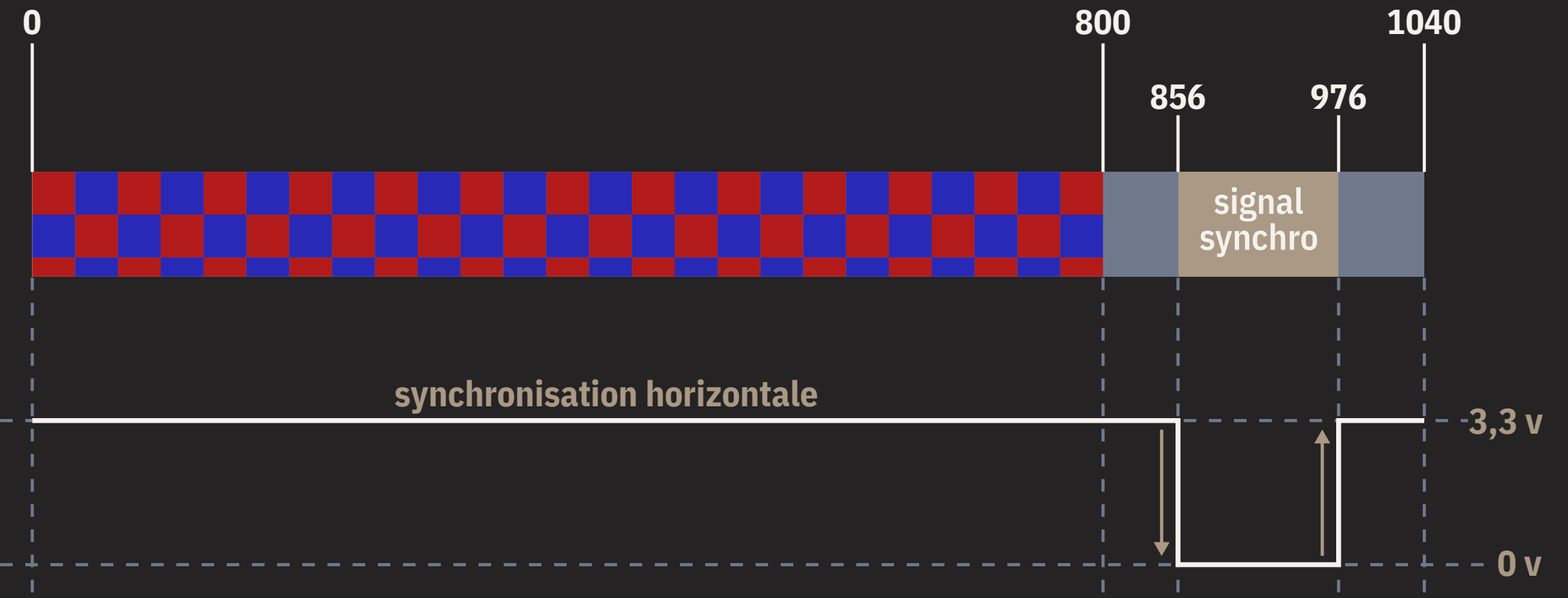

```
module SimpleVGA (
        input  wire clk,

        output wire hsync,
        output wire vsync,
        output reg  red,
        output reg  green,
        output reg  blue
);
```

CRÉATION DU CIRCUIT INTÉGRÉ, SYNCHRONISATION


```
module SimpleVGA (
        input  wire clk,

        output wire hsync,
        output wire vsync,
        output reg  red,
        output reg  green,
        output reg  blue
);
```

CRÉATION DU CIRCUIT INTÉGRÉ, COULEURS

SYNCHRONISATION HORIZONTALE



DÉCLENCHEMENT DE LA SYNCHRONISATION HORIZONTALE

```
reg [10:0] xpos = 0;
always @(posedge clk)
        if (xpos == 1039) xpos <= 0;
        else              xpos <= xpos + 1;

assign hsync = xpos < 856 || xpos >= 976;
```

CODE GÉNÉRANT LA SYNCHRONISATION HORIZONTALE

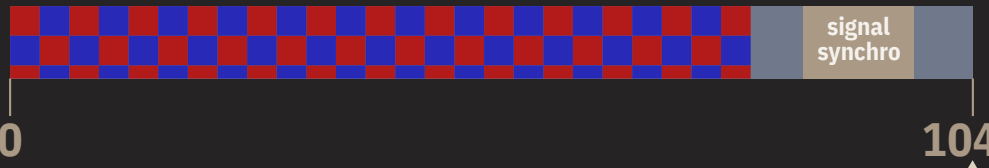
registre de 11 bits = valeurs de 0 à 2047

```
reg [10:0] xpos = 0;
always @(posedge clk)
        if (xpos == 1039) xpos <= 0;
        else              xpos <= xpos + 1;

assign hsync = xpos < 856 || xpos >= 976;
```

DÉCLARATION D'UN REGISTRE DE 11 BITS


```
reg [10:0] xpos = 0;
always @(posedge clk)
        if (xpos == 1039) xpos <= 0;
        else              xpos <= xpos + 1;

assign hsync = xpos < 856 || xpos >= 976;
```

EXÉCUTION À CHAQUE FRONT MONTANT




```
reg [10:0] xpos = 0;
always @(posedge clk)
        if (xpos == 1039) xpos <= 0;
        else              xpos <= xpos + 1;

assign hsync = xpos < 856 || xpos >= 976;
```

REMISE À ZÉRO APRÈS 1040 POINTS

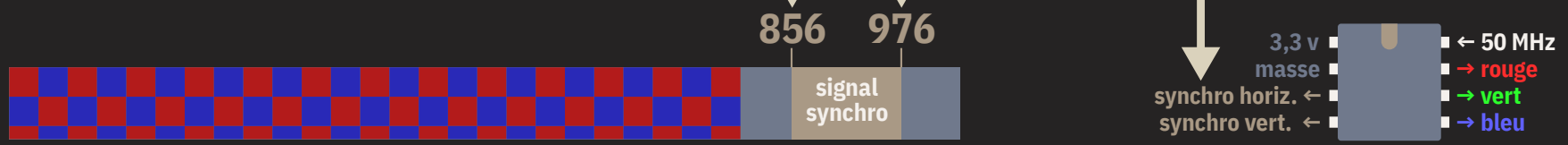

```
reg [10:0] xpos = 0;
always @(posedge clk)
        if (xpos == 1039) xpos <= 0;
        else              xpos <= xpos + 1;

assign hsync = xpos < 856 || xpos >= 976;
```

COMPTAGE DE 0 À 1039

exécution continue

exécution rythmée par une horloge

```
reg [10:0] xpos = 0;
always @(posedge clk)
        if (xpos == 1039) xpos <= 0;
        else              xpos <= xpos + 1;

assign hsync = xpos < 856 || xpos >= 976;
```



GÉNÉRATION DE LA SYNCHRONISATION HORIZONTALE

SYNCHRONISATION VERTICALE

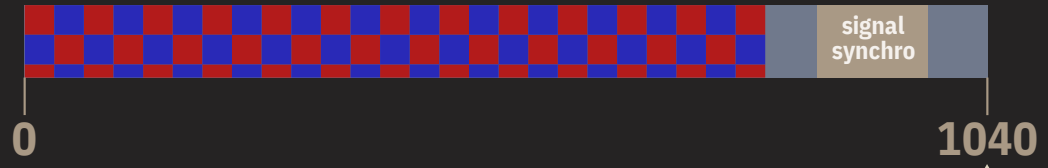
```
reg [9:0] ypos = 0;
always @(posedge clk)
        if (xpos == 1039) begin
                if (ypos == 665) ypos <= 0;
                else             ypos <= ypos + 1;
        end

assign vsync = ypos < 637 || ypos >= 643;
```

CODE GÉNÉRANT LA SYNCHRONISATION VERTICALE

registre de 10 bits = valeurs de 0 à 1023

```
reg [9:0] ypos = 0;
always @(posedge clk)
        if (xpos == 1039) begin
                if (ypos == 665) ypos <= 0;
                else             ypos <= ypos + 1;
        end

assign vsync = ypos < 637 || ypos >= 643;
```

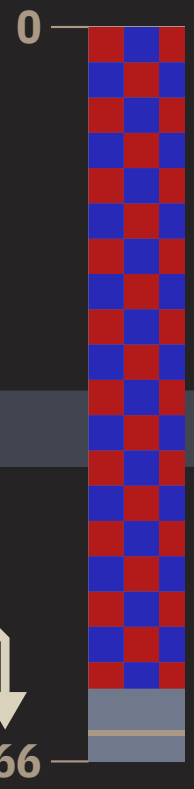
DÉCLARATION D'UN REGISTRE DE 10 BITS




```
reg [9:0] ypos = 0;
always @(posedge clk)
        if (xpos == 1039) begin
                if (ypos == 665) ypos <= 0;
                else             ypos <= ypos + 1;
        end

assign vsync = ypos < 637 || ypos >= 643;
```

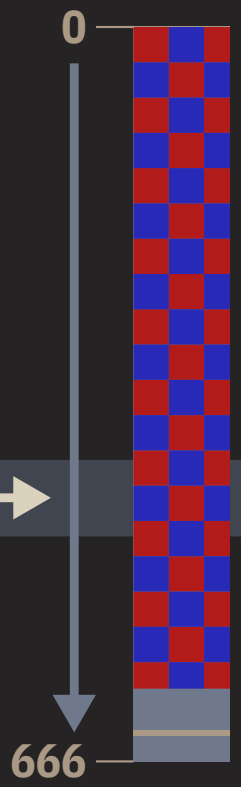
CALCUL À CHAQUE FIN DE LIGNE

```
reg [9:0] ypos = 0;
always @(posedge clk)
        if (xpos == 1039) begin
                if (ypos == 665) ypos <= 0;
                else             ypos <= ypos + 1;
        end

assign vsync = ypos < 637 || ypos >= 643;
```



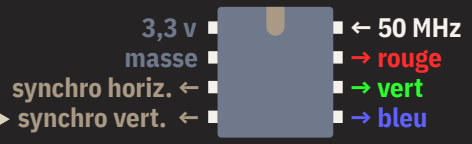
REMISE À ZÉRO APRÈS 666 LIGNES

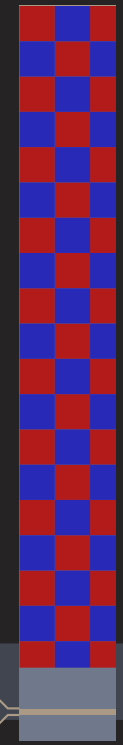
```
reg [9:0] ypos = 0;
always @(posedge clk)
        if (xpos == 1039) begin
                if (ypos == 665) ypos <= 0;
                else             ypos <= ypos + 1;
        end

assign vsync = ypos < 637 || ypos >= 643;
```



COMPTAGE DE 0 À 665

```
reg [9:0] ypos = 0;
always @(posedge clk)
        if (xpos == 1039) begin
                if (ypos == 665) ypos <= 0;
                else             ypos <= ypos + 1;
        end

assign vsync = ypos < 637 || ypos >= 643;
```





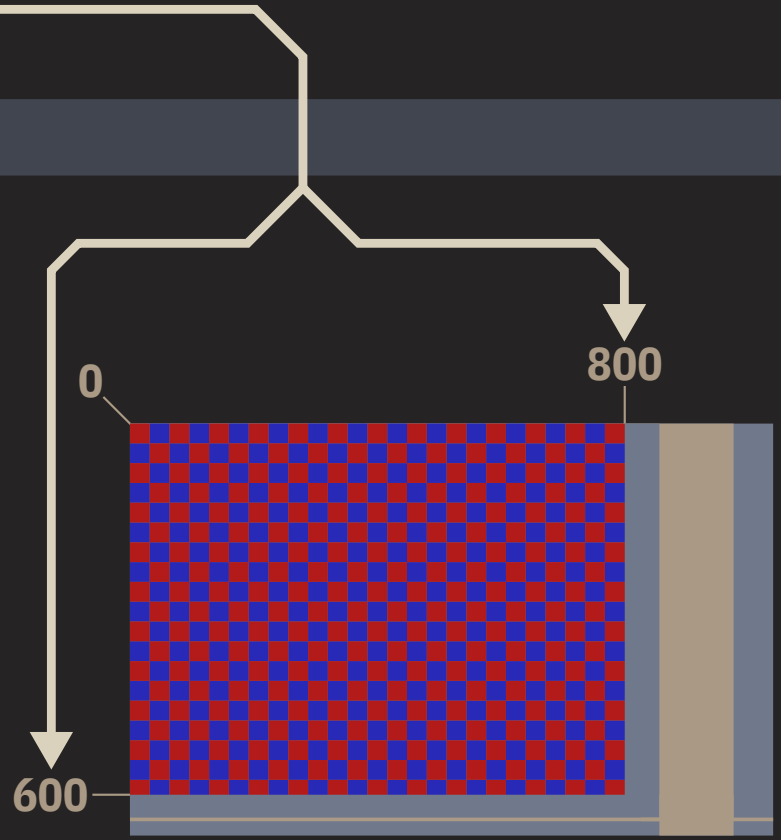
GÉNÉRATION DE LA SYNCHRONISATION VERTICALE

DAMIER ROUGE ET BLEU

```
always @(posedge clk)
        if (xpos < 800 && ypos < 600) begin
                red   <= ~xpos[5] ^ ypos[5];
                green <= 0;
                blue  <=  xpos[5] ^ ypos[5];
        end else begin
                red   <= 0;
                green <= 0;
                blue  <= 0;
        end
```

CODE GÉNÉRANT LE DAMIER ROUGE ET BLEU

```
always @(posedge clk)
        if (xpos < 800 && ypos < 600) begin
                red   <= ~xpos[5] ^ ypos[5];
                green <= 0;
                blue  <=  xpos[5] ^ ypos[5];
        end else begin
                red   <= 0;
                green <= 0;
                blue  <= 0;
        end
```



TEST DE LA ZONE VISIBLE

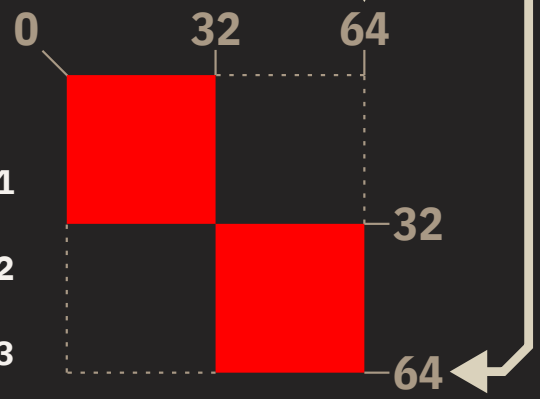
```
always @(posedge clk)
        if (xpos < 800 && ypos < 600) begin
                red   <= ~xpos[5] ^ ypos[5];
                green <= 0;
                blue  <=  xpos[5] ^ ypos[5];
        end else begin
                red   <= 0;
                green <= 0;
                blue  <= 0;
        end
```

les 3 calculs et affectations sont exécutés en parallèle

UN FPGA TRAVAILLE EN PARALLÈLE

```
always @(posedge clk)
        if (xpos < 800 && ypos < 600) begin
                red   <= ~xpos[5] ^ ypos[5];
                green <= 0;
                blue  <=  xpos[5] ^ ypos[5];
        end else begin
                red   <= 0;
                green <= 0;
                blue  <= 0;
        end
```

10	9	8	7	6	**5**	4	3	2	1	0	
0	0	0	0	0	**0**	0	0	0	0	0	= 0
0	0	0	0	0	**0**	1	1	1	1	1	= 31
0	0	0	0	0	**1**	0	0	0	0	0	= 32
0	0	0	0	0	**1**	1	1	1	1	1	= 63

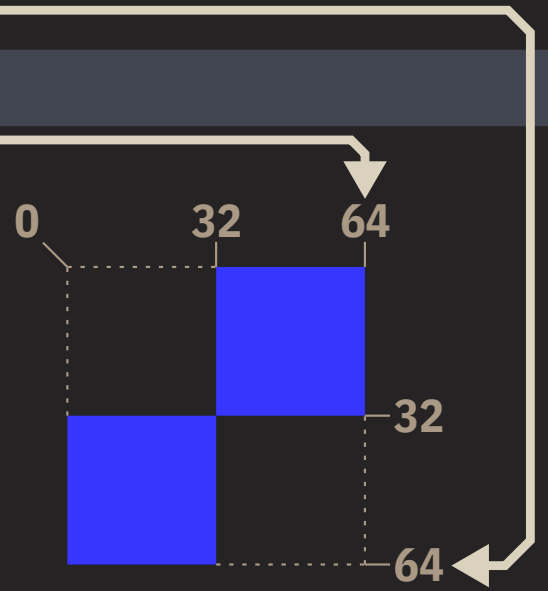


ALTERNER LES CARRÉS ROUGES

```
always @(posedge clk)
        if (xpos < 800 && ypos < 600) begin
                red   <= ~xpos[5] ^ ypos[5];
                green <= 0;
                blue  <=  xpos[5] ^ ypos[5];
        end else begin
                red   <= 0;
                green <= 0;
                blue  <= 0;
        end
```

LE VERT N'EST JAMAIS UTILISÉ

```
always @(posedge clk)
        if (xpos < 800 && ypos < 600) begin
                red   <= ~xpos[5] ^ ypos[5];
                green <= 0;
                blue  <=  xpos[5] ^ ypos[5];
        end else begin
                red   <= 0;
                green <= 0;
                blue  <= 0;
        end
```



ALTERNER LES CARRÉS BLEUS

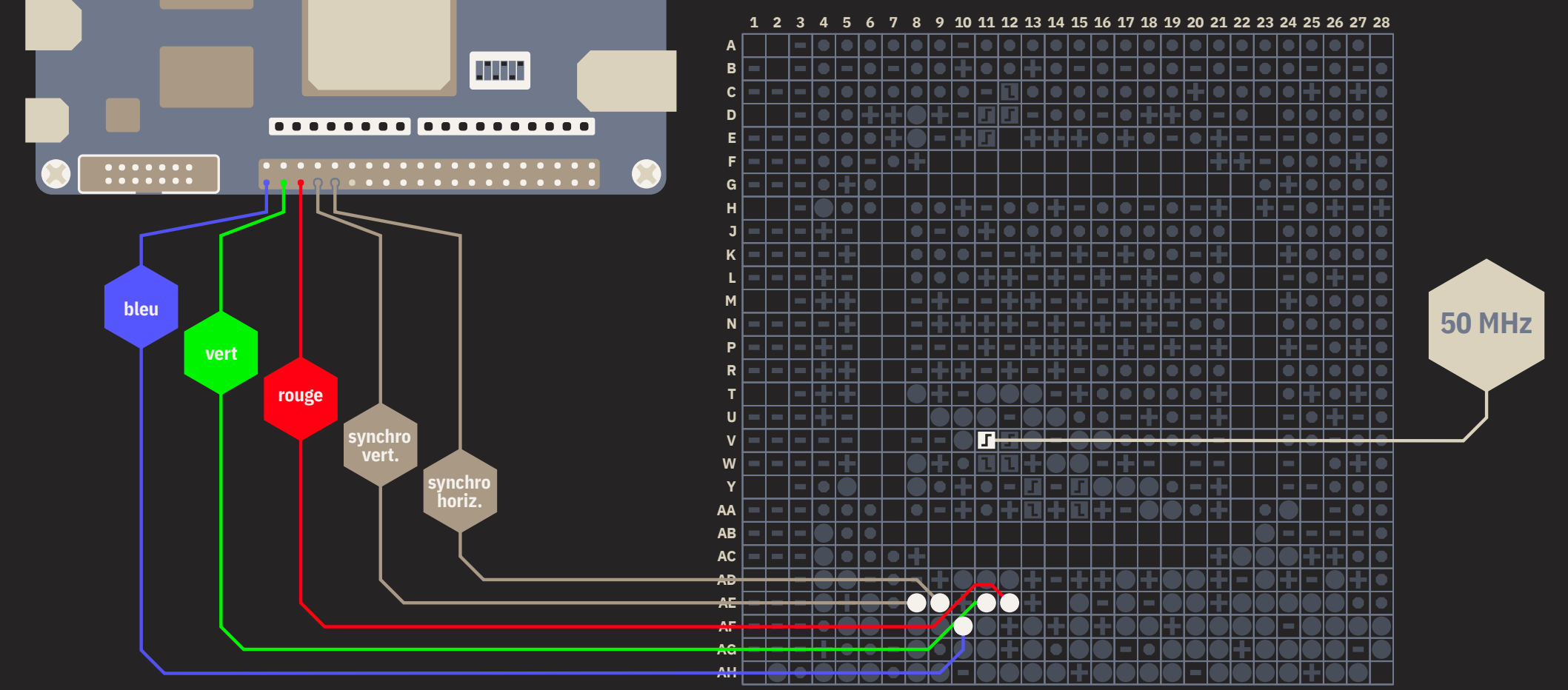
```
always @(posedge clk)
        if (xpos < 800 && ypos < 600) begin
                red   <= ~xpos[5] ^ ypos[5];
                green <= 0;
                blue  <=  xpos[5] ^ ypos[5];
        end else begin
                red   <= 0;
                green <= 0;
                blue  <= 0;
        end
```

AUCUNE COULEUR EN DEHORS DE LA ZONE VISIBLE

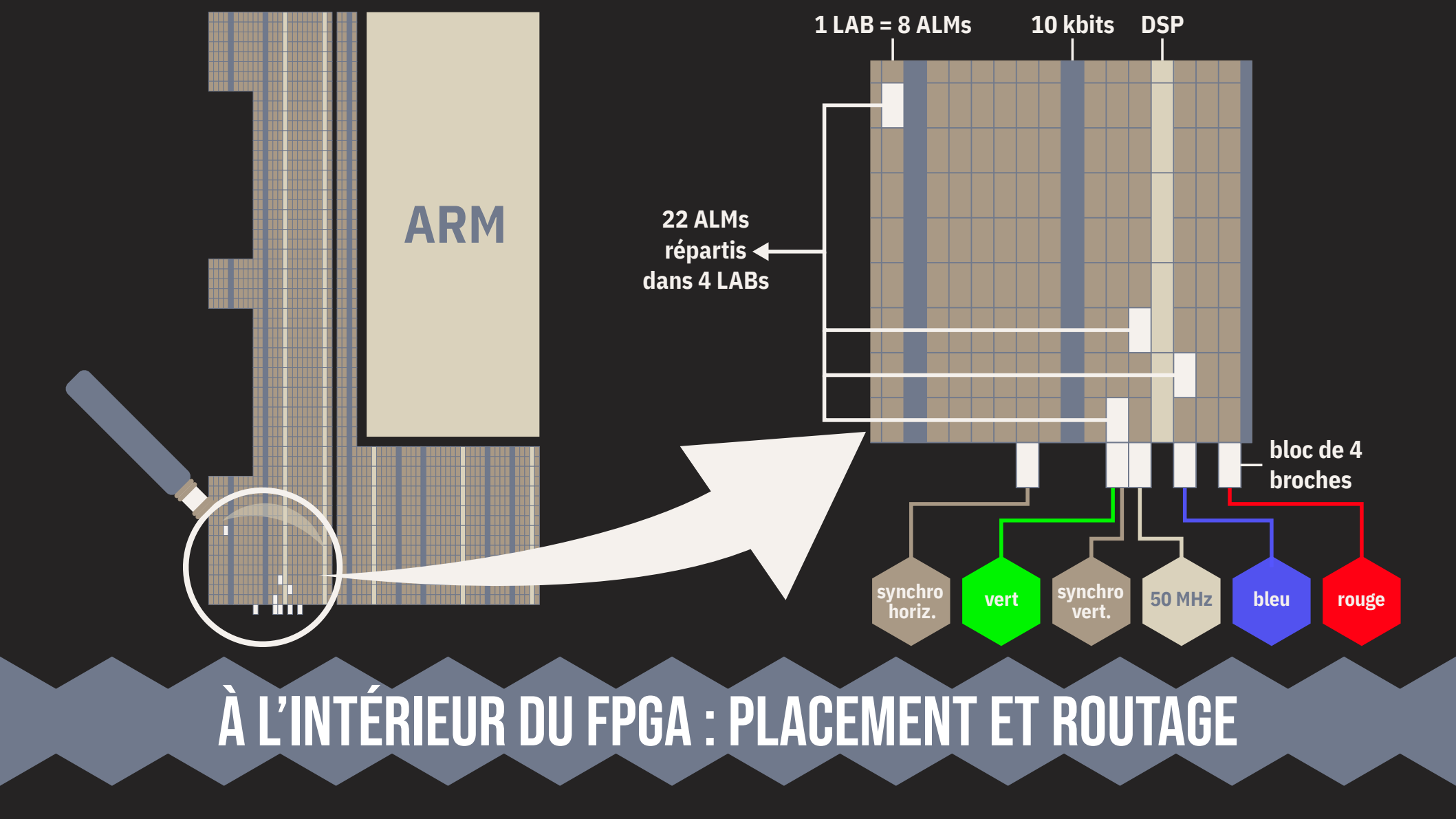
Remarques et pièges

- Verilog n’est **pas un langage procédural**
- Un bit peut avoir 4 états
  - 0, 1, X et Z → faux, vrai, inconnu, impédance haute
- Vous êtes responsables des délais de propagation
- Générer une image « bitstream » est long…
- Tous les codes HDL ne sont pas synthétisables

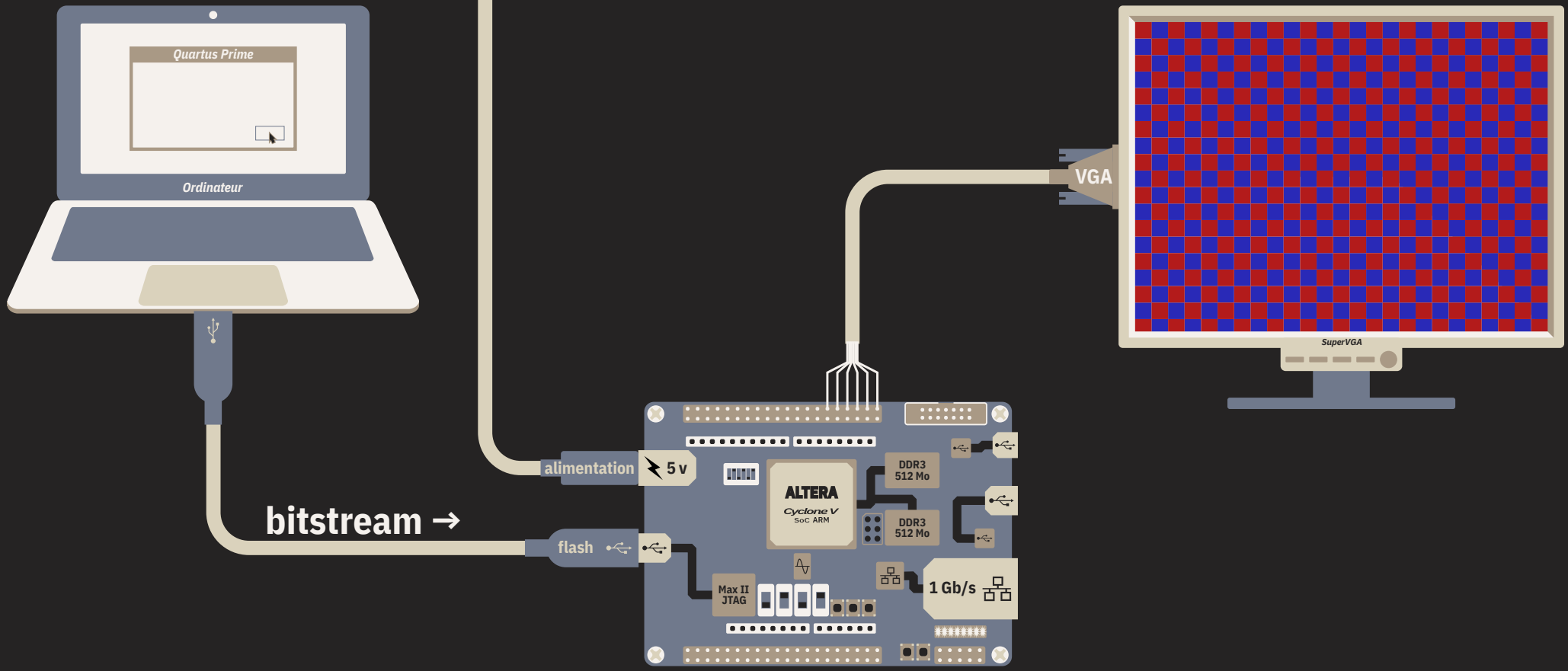
GÉNÉRATION DE L'IMAGE « BITSTREAM »



À L'EXTÉRIEUR DU FPGA : AFFECTATION DES BROCHES

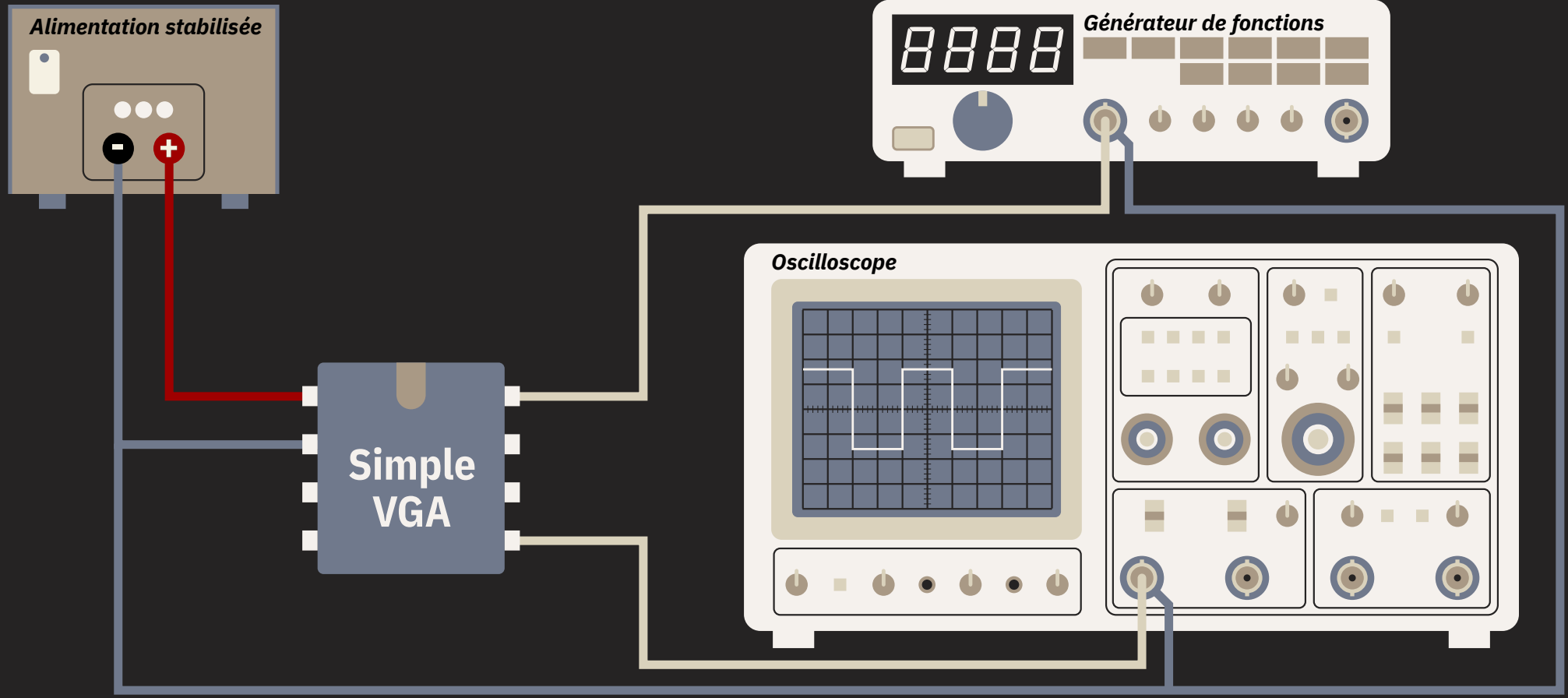


À L'INTÉRIEUR DU FPGA : PLACEMENT ET ROUTAGE



ENVOI DE L'IMAGE « BITSTREAM »

DÉBOGUER UN CIRCUIT



DÉBOGUER PHYSIQUEMENT

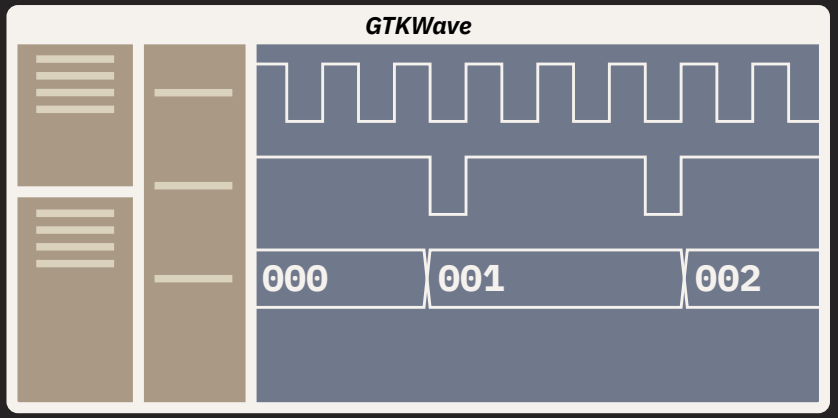
Icarus Verilog

```
module SimpleVGA (
        input wire clk,

        output wire hsync,
        output wire vsync,
        output reg red,
        output reg green,
        output reg blue
);

reg [10:0] xpos = 0;
always @(posedge clk)
        if (xpos == 1039) xpos <= 0;
        else              xpos <= xpos + 1;

assign hsync = xpos < 856 || xpos >= 976;

reg [9:0] ypos = 0;
always @(posedge clk)
        if (xpos == 1039)
                if (ypos == 665) ypos <= 0;
                else             ypos <= ypos + 1;

assign vsync = ypos < 637 || ypos >= 643;

always @(posedge clk)
        if (xpos < 800 && ypos < 600) begin
                red <= xpos[6] ^ ypos[6];
                green <= 0;
                blue <= ~xpos[5] ^ ypos[5];
        end else begin
                red <= 0;
                green <= 0;
                blue <= 0;
        end

endmodule
```

```
`timescale 10ns / 10ns
module SimpleVGA_tb;
        reg clk = 0;
        always #1 clk = !clk;

        wire hsync;

        SimpleVGA testbench(
                .clk (clk),
                .hsync (hsync)
        );

        initial begin
                $dumpfile("SimpleVGA_tb.vcd");
                $dumpvars(0, testbench);

                #1385280 $finish;
        end

        reg [24:0] counter = 0;
        always @(negedge hsync) begin
                counter = counter + 1;
                $display("HSYNC = %0d @ %0t ns", counter, $time);
        end
endmodule
```

SimpleVGA + **Banc d'essai**

GTKWave



SIMULER POUR DÉBOGUER

ORGANISATION DU BANC D'ESSAI

1) Génération d'une horloge
2) Instanciation de SimpleVGA
3) Lancement de la simulation
4) Création d'un compteur

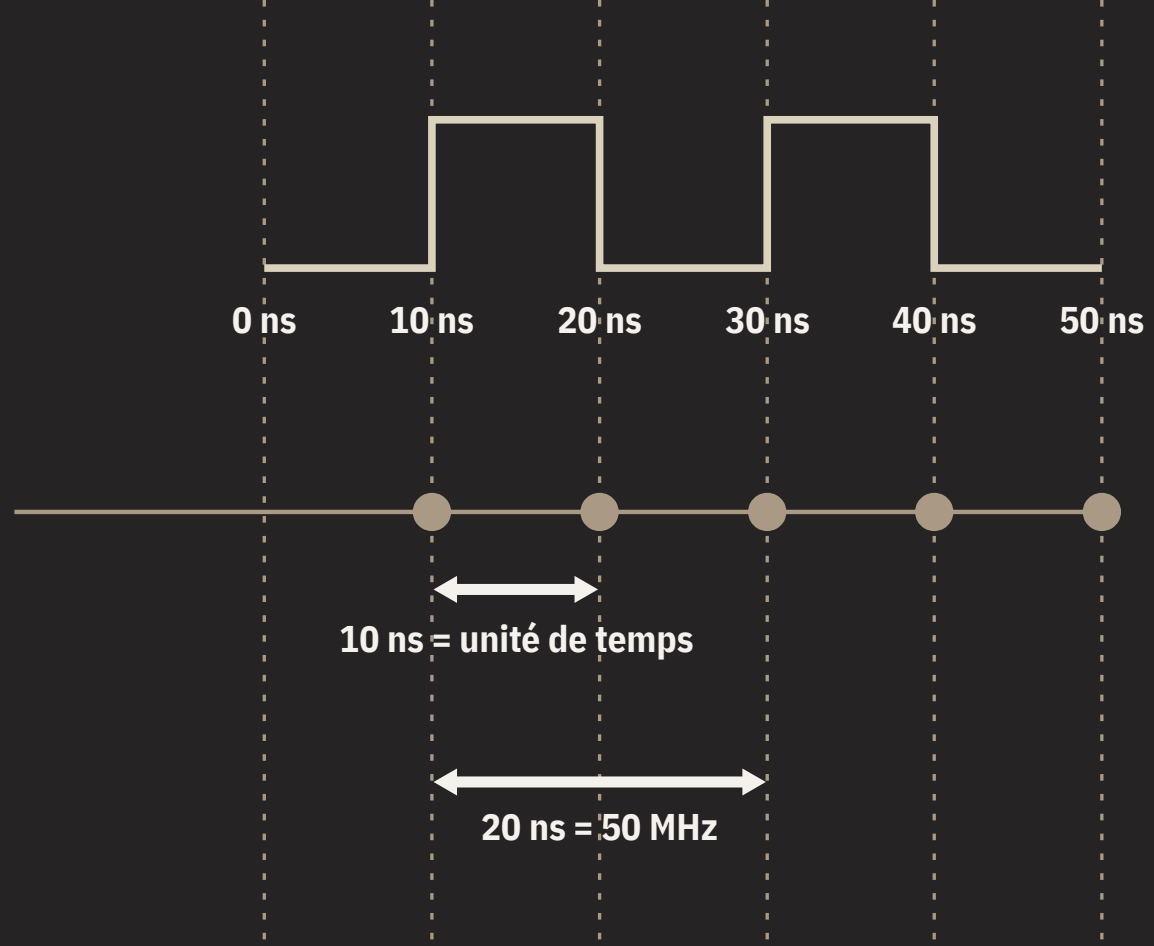
1
```
`timescale 10ns / 10ns
module SimpleVGA_tb;
        reg clk = 0;
        always #1 clk = !clk;

        wire hsync;
```

2
```
        SimpleVGA testbench(
                .clk (clk),
                .hsync (hsync)
        );
```

3
```
        initial begin
                $dumpfile("SimpleVGA_tb.vcd");
                $dumpvars(0, testbench);

                #1385280 $finish;
        end
```

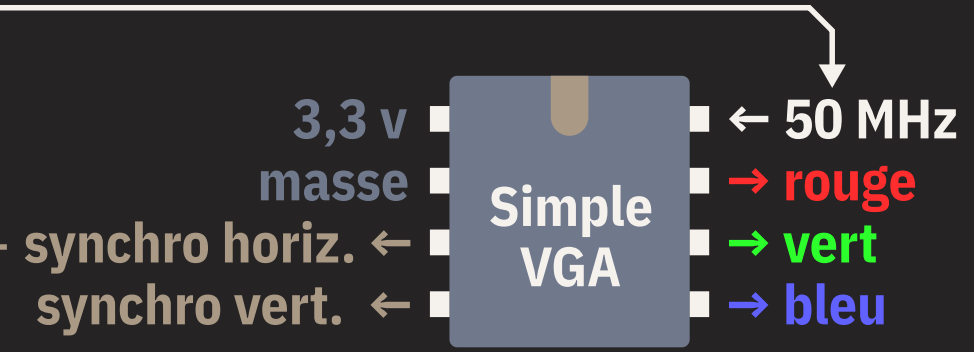
4
```
        reg [24:0] counter = 0;
        always @(negedge hsync) begin
                counter = counter + 1;
                $display("HSYNC = %0d @ %0t ns", counter, $time);
        end
endmodule
```

```
`timescale 10ns / 10ns
module SimpleVGA_tb;
        reg clk = 0;
        always #1 clk = !clk;

        wire hsync;
```



GÉNÉRATION D'UNE HORLOGE

```
SimpleVGA testbench(
        .clk (clk),
        .hsync (hsync)
);
```



INSTANCIATION DE SIMPLEVGA

```
initial begin
        $dumpfile("SimpleVGA_tb.vcd");
        $dumpvars(0, testbench);

        #1385280 $finish;
end
```

bloc exécuté une seule fois

tâche système

sauve tous les signaux de testbench

termine la simulation après
20 ns × 1040 × 666 = 13 852 800 ns

$$\simeq \frac{1}{72}\ \text{s}$$

LANCEMENT DE LA SIMULATION

```
reg [24:0] counter = 0;
always @(negedge hsync) begin
        counter = counter + 1;
        $display("HSYNC = %0d @ %0t ns", counter, $time);
end
```

CRÉATION D'UN COMPTEUR

Ligne de commande

```
$ iverilog -o SimpleVGA_tb SimpleVGA_tb.v SimpleVGA.v
```

COMPILATION DU BANC D'ESSAI

Ligne de commande

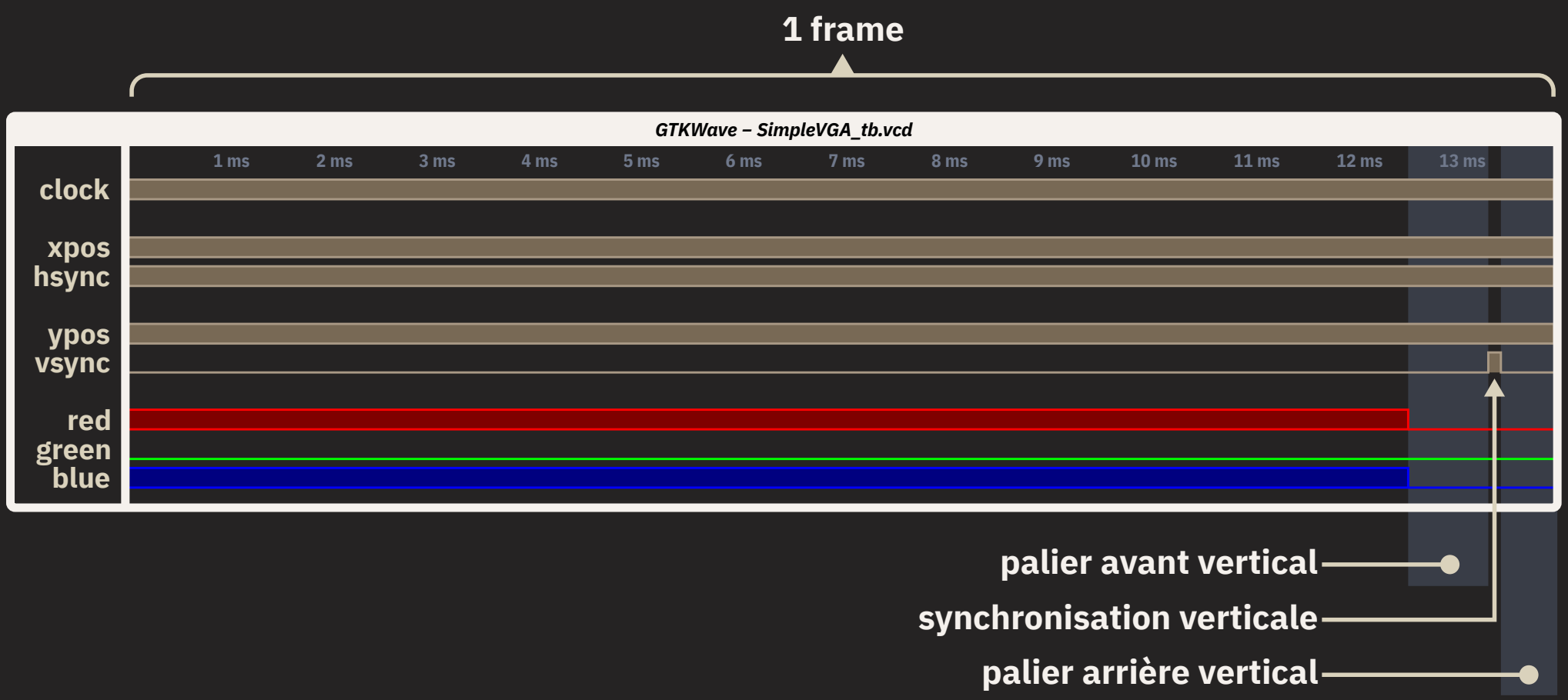
```
$ iverilog -o SimpleVGA_tb SimpleVGA_tb.v SimpleVGA.v
$ ./SimpleVGA_tb
VCD info: dumpfile SimpleVGA_tb.vcd opened for output.
HSYNC = 1 @ 1711 ns
HSYNC = 2 @ 3791 ns
HSYNC = 3 @ 5871 ns
# ...
HSYNC = 664 @ 1380751 ns
HSYNC = 665 @ 1382831 ns
HSYNC = 666 @ 1384911 ns
```

EXÉCUTION DE LA SIMULATION

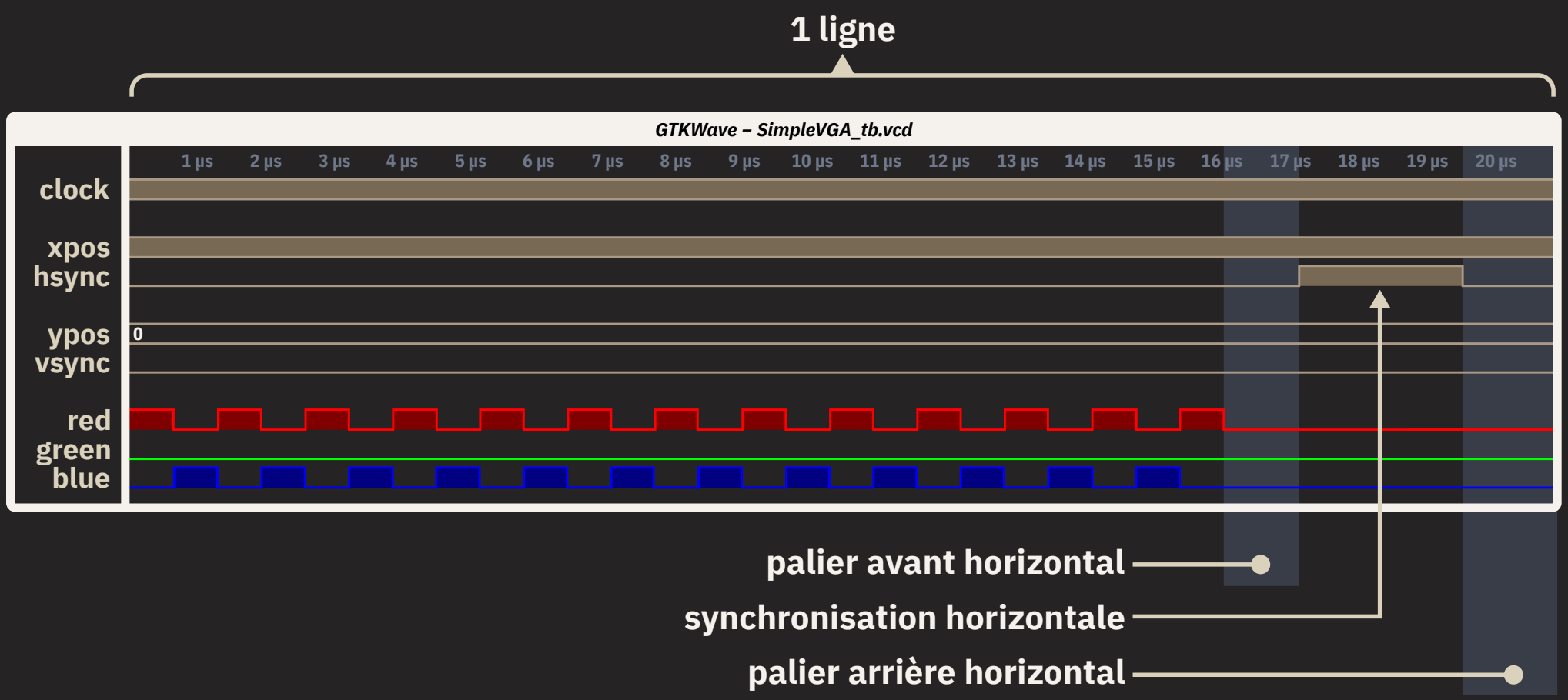
Ligne de commande

```
$ iverilog -o SimpleVGA_tb SimpleVGA_tb.v SimpleVGA.v
$ ./SimpleVGA_tb
VCD info: dumpfile SimpleVGA_tb.vcd opened for output.
HSYNC = 1 @ 1711 ns
HSYNC = 2 @ 3791 ns
HSYNC = 3 @ 5871 ns
# ...
HSYNC = 664 @ 1380751 ns
HSYNC = 665 @ 1382831 ns
HSYNC = 666 @ 1384911 ns
$ gtkwave SimpleVGA_tb.vcd
```

LANCEMENT DE GTKWAVE

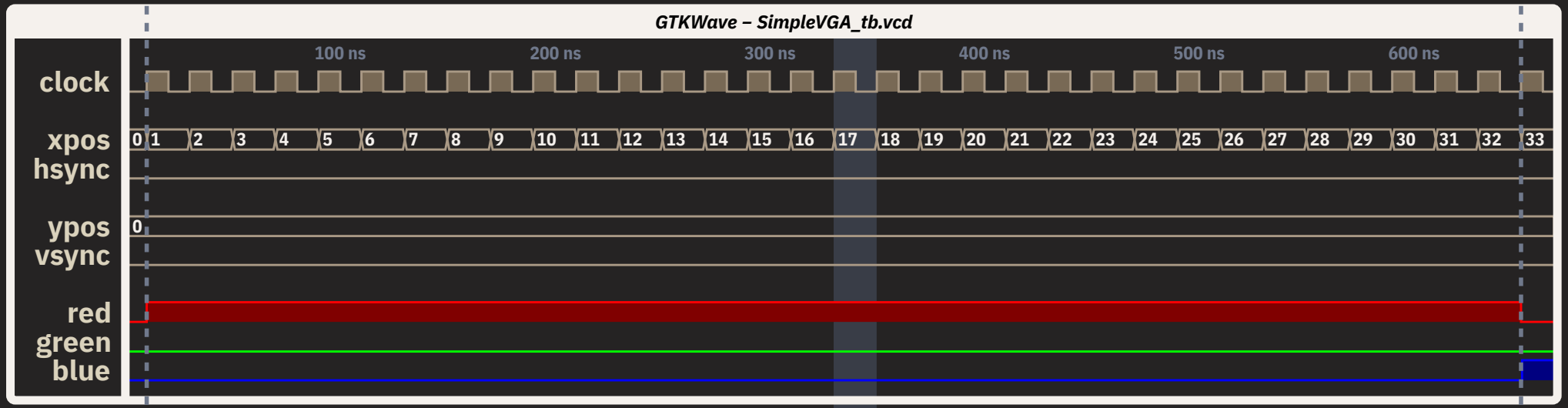


ÉVOLUTION DES SIGNAUX (FRAME)



ÉVOLUTION DES SIGNAUX (LIGNE)

32 cycles

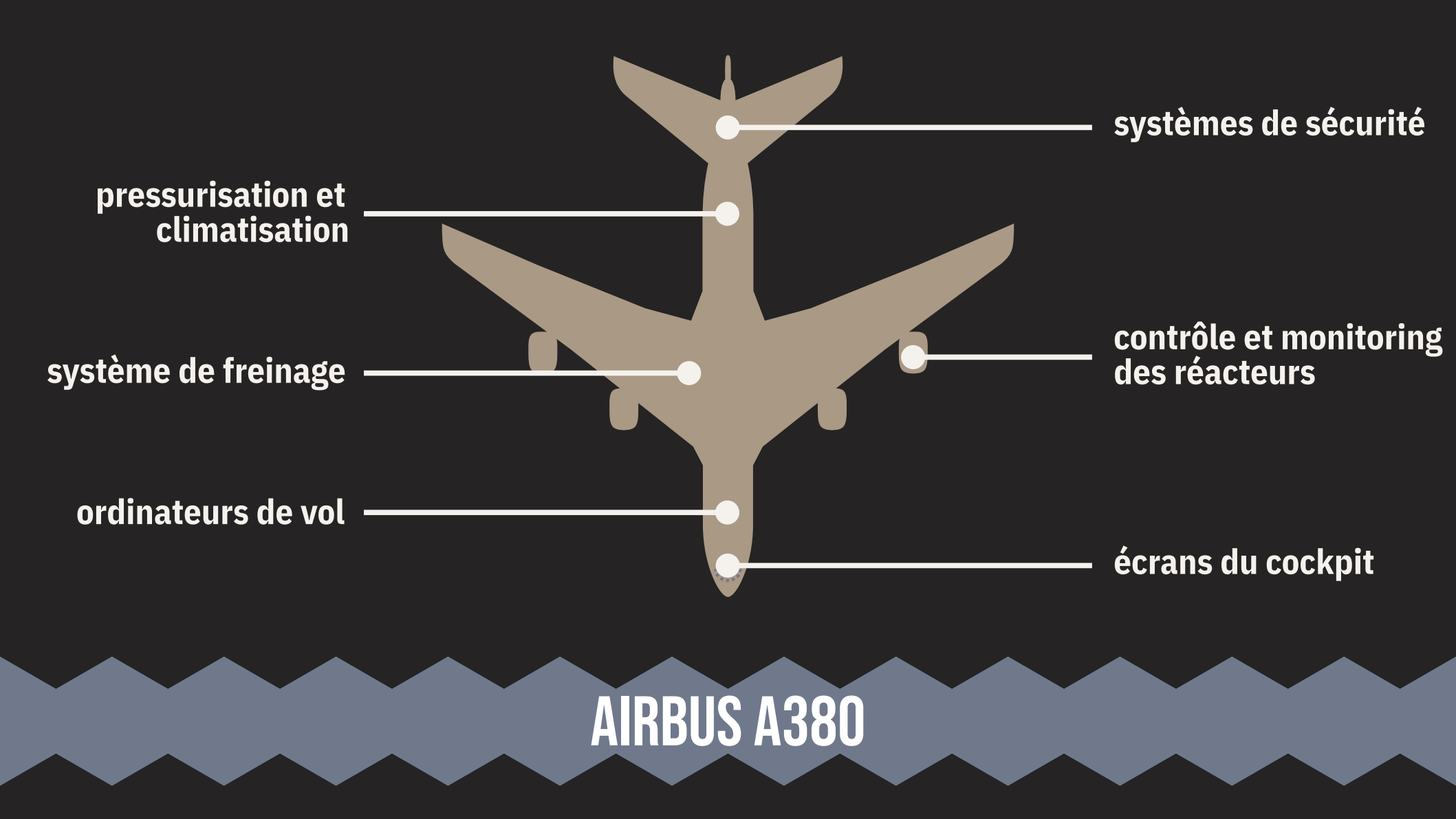


1 cycle = 20 ns / 50 MHz = 1 pixel

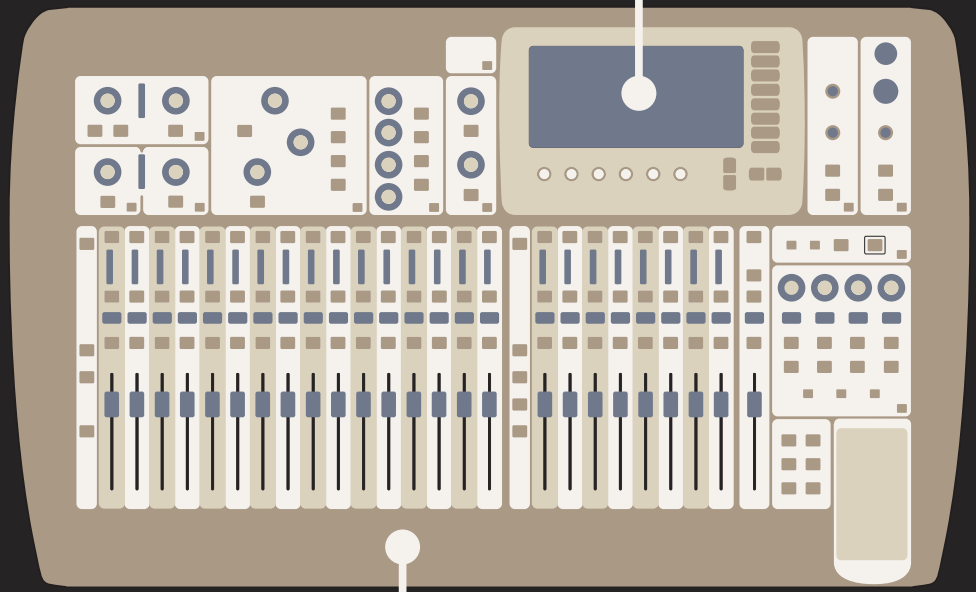
ÉVOLUTION DES SIGNAUX (HORLOGE)

ET MAINTENANT ?

QUELQUES EXEMPLES D'UTILISATION


systèmes de sécurité
pressurisation et climatisation
contrôle et monitoring des réacteurs
système de freinage
ordinateurs de vol
écrans du cockpit
AIRBUS A380

effets en temps réel

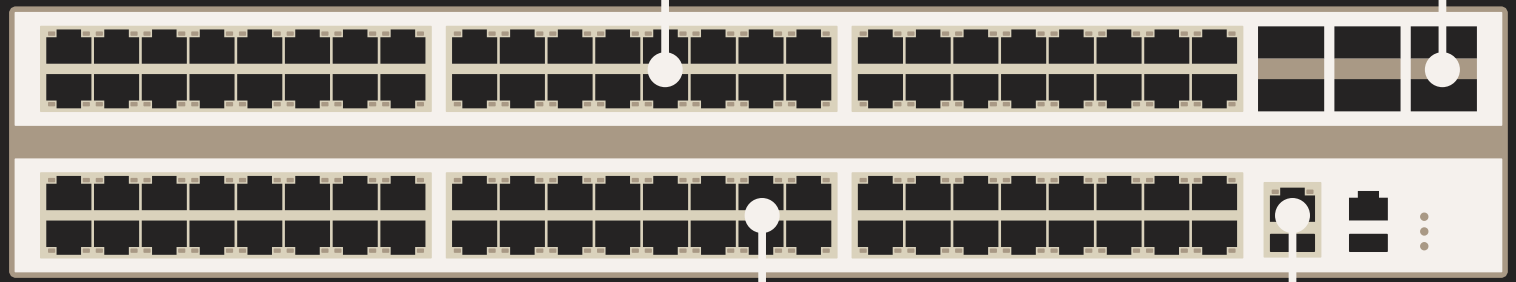


mixage de nombreuses entrées

TABLE DE MIXAGE BEHRINGER X32

très faible latence

très haut débit



gestion de nombreuses entrées

mises à jour

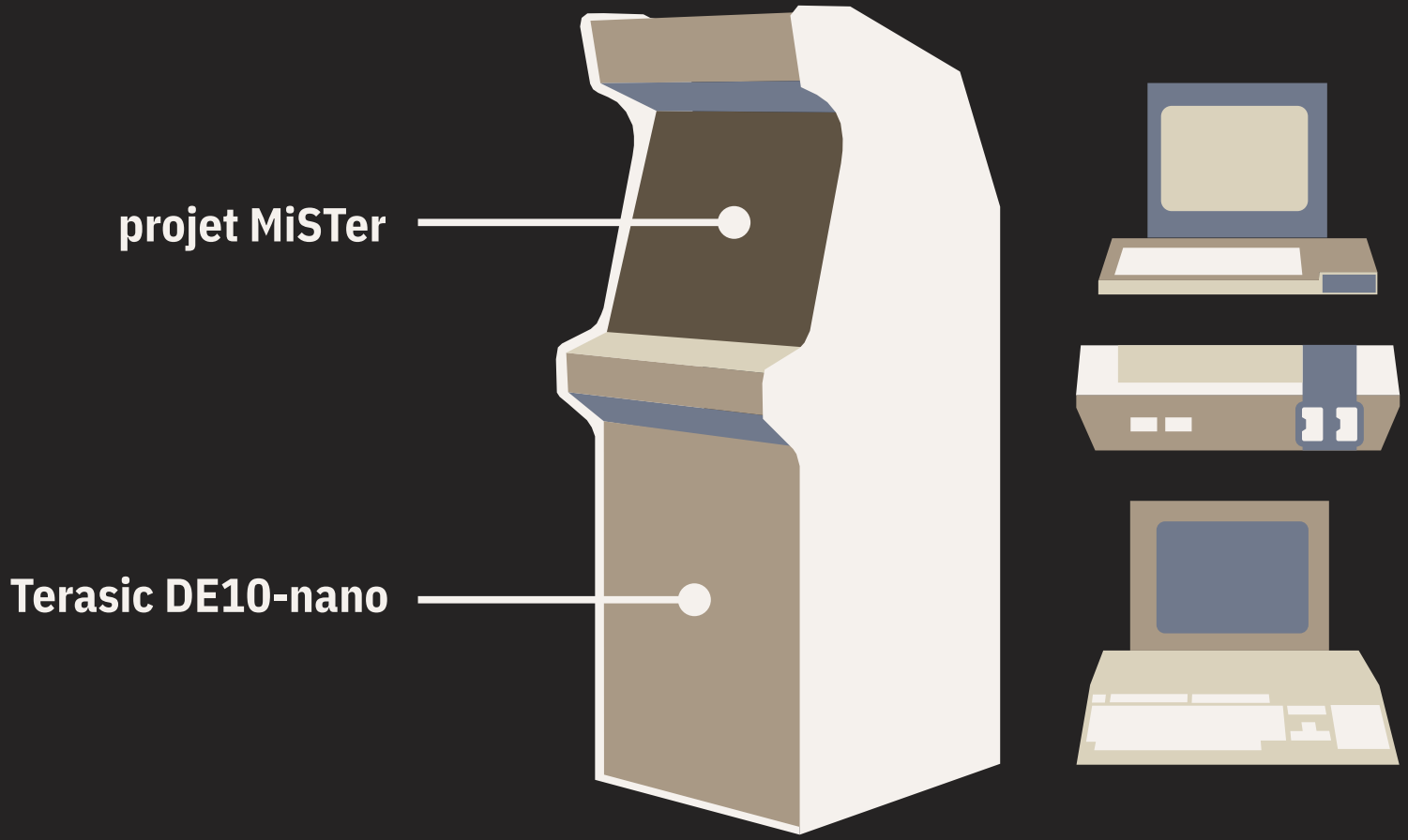
SWITCH CISCO

traitement d’image
après acquisition

traitement des
données brutes

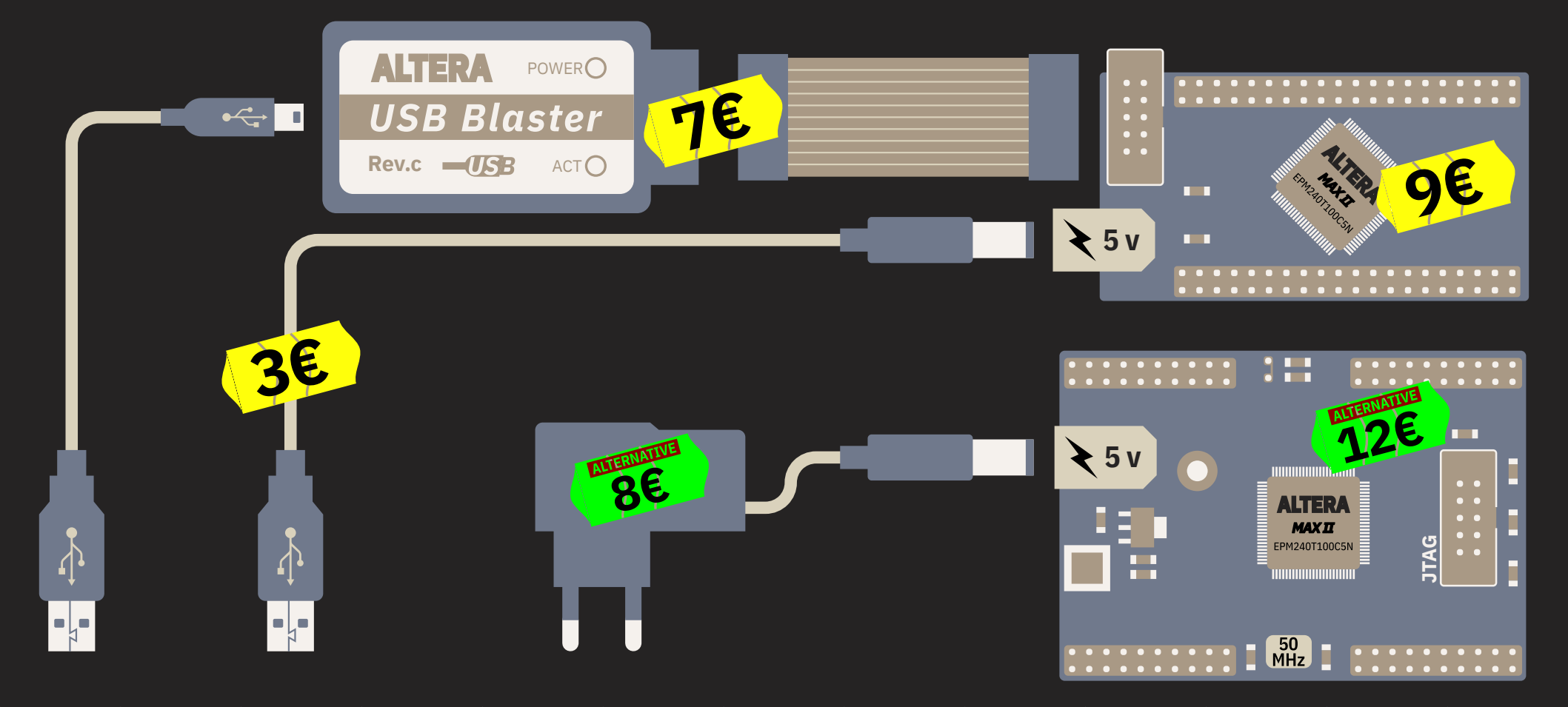
IMAGERIE MÉDICALE : IRM


SYN flood
UDP flood
TCP flood
VAC - VAC - VAC - VAC - VAC - VAC - VAC - VAC - VAC - VAC - VAC - VAC - VAC - VAC - VAC - VAC -
OVHcloud
Intel Stratix V
Xilinx Virtex UltraScale+
SYSTÈME ANTI-DDOS D'OVHCLOUD



ÉMULATION FINE D'ANCIENNES PLATEFORMES

PAR OÙ COMMENCER ?



AVEC UN BUDGET DE 20€

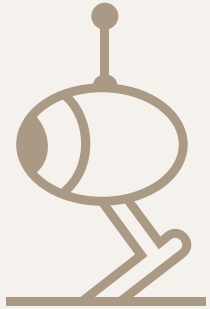
AVANTAGES/INCONVÉNIENTS DES FPGA

Inconvénients des FPGA

- Autre façon de programmer
  - paradigme
  - temps de compilation
  - débogage
- Positionnement
  - ASIC
  - GPU
  - microcontrôleur
- Ticket d'entrée
  - coût des logiciels
  - coût des circuits
- Écosystème limité
  - open source peu présent
  - disponibilité des compétences
  - OpenCL

AVANTAGES DES FPGA

- Plus proche de l’électron
  - flexibilité des entrées/sorties
  - moins de composants
  - consommation électrique
- Puissance de calcul
- Précision du signal
  - comportement déterministe
  - faible latence
- Reconfigurable
- Sécurité
  - pas de dépassement
  - offuscation ?



Merci de votre attention !

Merci à
Virginie Férey Rochefeuille, Rémi Passerieu, David Glaude, Thoma Hauc,
Tristan Groléat (OVHcloud), Francis Trautmann, Sébastien Dupire,
Loïc “Iooner”, Pascale Lambert-Charreteur,
Échelle Inconnue et l’équipe du Devfest Nantes

LICENCES

- Présentation sous Licence CC-BY 4.0
  - textes, images, gabarits… sauf mention contraire
  - https://creativecommons.org/licenses/by/4.0/legalcode.fr
- Les polices suivantes ont été utilisées
  - IBM Plex™, licence OFL
  - Bebas Neue, licence OFL

LOGICIELS

- Logiciels
  - Intel® Quartus® Prime Lite
  - ModelSim ASE
  - Icarus Verilog GPL
  - GTKWave GPL
- Présentation
  - Inkscape GPL Free Software Free as in Freedom
  - LibreOffice Impress MPL V2.0
  - LibreOffice Writer MPL V2.0
  - LibreOffice Calc MPL V2.0
  - JabRef MIT
  - hilite.me FREEWARE
  - Coolors.co FREEWARE
  - TinyVGA FREEWARE

BIBLIOGRAPHIE

BIBLIOGRAPHIE 1

- **Moore, Gordon Earle**
  Cramming more components onto integrated circuits
  Semiconductor, Fairchild
  1965

- **Sugarman, Robert**
  Does the country need a good $20 microprocessor?
  1975
- **Moore, Gordon Earle**
  Progress in digital integrated electronics
  Intel
  1975
- **Waite, Mitchell**
  Computer Graphics Primer
  Howard W. Sams \& Co., Inc.
  1979
- **Veen, Arthur H.**
  Dataflow machine architecture
  Surveys, A. C. M. Computing
  1986
- **Freeman, Ross H.**
  Configurable electrical circuit having configurable logic elements and configurable interconnects
  1989
- **Kahng, A. B. & Pati, Y. C.**
  Subwavelength optical lithography, challenges and impact on physical design
  of Computer Science, U. C. L. A. Department & Numerical Technologies, Inc
  1999

BIBLIOGRAPHIE 2

- **Schüler, E. & Helfers, Tim**
  XPP - eXtreme Processing Platform Technology for space applications
  Astrium, P. A. C. T.
  2001
- **Zoltan Baruch, Octavian Creț & Pusztai, Kalman**
  Configurable processor
  of Cluj-Napoca, Technical University
  2002
- **Compton, Katherine & Hauck, Scott**
  Reconfigurable computing: a survey of systems and software
  Surveys, A. C. M. Computing
  2002
- **Gonzalez, Ricardo E.**
  A Software Configurable Processor
  Inc, Stretch
  2005
- **WEBER, Charles; BERGLUND, C. Neil & GABELLA, Patricia**
  Mask cost and profitability in photomask manufacturing, an empirical analysis
  University, Portland State
  2006
- **Carton, Olivier**
  Transistors et portes logiques
  de Recherche en Informatique Fondamentale, Institut
  2006

BIBLIOGRAPHIE 3

- **Prado, Daniel Francisco Gómez**
  Tutorial on FPGA routing
  of Massachusetts, University
  2006
- **Unknown**
  6502 Schematic
  Unknown
  2007
- **Museum, Computer History**
  Oral history panel on the development and promotion of the Motorola 68000
  2007
- **Drepper, Ulrich**
  What every programmer should know about memory, Part 1
  LWN
  2007
- **Weisberg, David E.**
  The Engineering Design Revolution
  2008
- **Wayne Wolf, Ahmed Amine Jerraya & Martin, Grant**
  Multiprocessor System-on-Chip (MPSoC) Technology
  IEEE
  2008
- **Articles, A. R. M. Technical Support Knowledge**
  What is the difference between a von Neumann architecture and a Harvard architecture?
  Limited, A. R. M.
  2008
- **Fouquet-Lapar, Matthias**
  The von Neumann Architecture and Alternatives
  SGI
  2008

BIBLIOGRAPHIE 4

- **Museum, Computer History**
  Altera EP300 Design \& Development Oral History Panel
  2009
- **McGrath, Dylan**
  FPGA startups stare down giants and ghosts
  Times, Electronic Engineering
  2009
- **Jeff Chase, Brent Nelson, John Bodily Zhaoyi Wei & Lee, Dah-Jye**
  Real-Time Optical Flow Calculations on FPGA and GPU Architectures: AComparison Study
  University, Brigham Young
  2009
- **Kidd, Taylor IoT**
  *Why P scales a C*V^2*f is so obvious*
  Zone, Intel Developer
  2009
- **Conrad**
  Kit d'apprentissage de l'électronique pour débutants
  2009
- **Jones, David H.; Powell, Adam; Bouganis, Christos-Savvas & Cheung, Peter Y. K.**
  GPU versus FPGA for high productivity computing
  London, Imperial College
  2010

BIBLIOGRAPHIE 5

- **Kalarot, Ratheesh & Morris, John**
  Comparison of FPGA and GPU implementations of Real-time Stereo Vision
  of Auckland / IEEE, The University
  2010
- **Cox, Russ**
  The MOS 6502 and the Best Layout Guy in the World
  2011
- **Culver, John**
  How a CPU Microprocessor is made
  Shack, The C. P. U.
  2011
- **Feugey, David**
  Altera mise sur l'OpenCL pour révolutionner le monde des FPGA
  2011
- **Johnson, Jeff**
  Outsourcing FPGA design: pros and cons
  2011
- **VERRY, Tim**
  Apple's A6 processor uses hand drawn ARM cores to boost performance
  Perspective, P. C.
  2012
- **Nenni, Daniel**
  A Brief History of FPGAs
  2012
- **persons, Various**
  What are some examples of non-Von Neumann architectures?
  StackOverflow
  2012

BIBLIOGRAPHIE 6

- **Fowers, Jeremy; Brown, Greg; Cooke, Patrick & Stitt, Greg**
  A performance and energy comparison of FPGAs, GPUs and Multicores for sliding-window applications
  of Florida, University
  2012
- *An FPGA-based supercomputer for statistical physics: the weird case of Janus*
  2012
- **Wikipedia**
  P.A. Semi
  2013
- **Electronics, Gould**
  Electrically Erasable Programmable Logic PEEL 18CV8
  2013
- **Altera**
  Implementing FPGA design with the OpenCL standard
  2013
- **Miller, Warren**
  Configurable processors as an alternative to FPGAs
  Times, Electronic Engineering
  2013
- **Higginbotham, Stacey**
  Why Microsoft is building programmable chips that specialize in search
  GigaOM
  2014
- **Hindriksen, Vincent**
  Why use OpenCL on FPGAs?
  StreamHPC
  2014

BIBLIOGRAPHIE 7

- **VAndrei**
  Von Neumann vs Harvard architecture
  StackOverflow
  2014
- **Jones, Dr. Handel**
  Why migration to 20 nm bulk CMOS and 16/14 nm FinFets is not best approach for semiconductor industry
  Strategies, International Business
  2014
- **McMillan, John**
  PCB design then and now
  Mentor, a Siemens Business
  2015
- **Franz, Kaitlyn**
  History of the FPGA
  Inc, Digilent
  2015
- **Intel**
  Intel completes acquisition of Altera
  2015
- **Higginbotham, Stacey**
  Why Intel will spend $16.7 billion on Altera
  Fortune
  2015
- **Harris, Derrick**
  Microsoft is building fast, low-power neural networks with FPGAs
  GigaOM
  2015

BIBLIOGRAPHIE 8

- **Ovtcharov, Kalin; Ruwase, Olatunji; Kim, Joo-Young; Fowers, Jeremy; Strauss, Karin & Chung, Eric S.**
  Accelerating deep convolutional neural networks using specialized hardware
  Research, Microsoft
  2015
- **et Technologie), U. N. I. T. (Université Numérique Ingénierie**
  Les grands mythes fondateurs" des nanos : la loi de Moore ou l'héritage du talk de Feynman de 1959"
  2015
- **Kidd, Taylor IoT**
  *Why P scales as C*V^2*f is so obvious (pt 2)*
  Zone, Intel Developer
  2015
- **Economist, The**
  The end of Moore's law
  2015
- **Dettmers, Tim**
  Is implementing deep learning on FPGAs a natural next step after the success with GPUs?
  2015
- **Ucamco, former Barco E. T. S.**
  Cilbr8tor Series
  2016
- **HKallaher, Brandon**
  PAL vs. CPLD vs. FPGA
  Blog, Digilent
  2016

BIBLIOGRAPHIE 9

- **Denisenko, Dmitry**
  OpenCL for FPGAs
  Intel
  2016
- **Eijkhout, Victor**
  Are there alternatives to the Von Neumann architecture?
  Quora
  2016
- **Vašut, Marek**
  Open-Source Tools for FPGA Development
  Engineering, D. E. N. X. Software & Foundation, The Linux
  2016
- **Economist, The**
  After Moore's law, the future of computing
  2016
- **Processing, Berten Digital Signal**
  GPU vs FPGA Performance Comparison (white paper)
  2016
- **Koch, Dirk; Hannig, Frank; Ziener, Daniel**
  FPGAs for Software Programmers
  Springer
  2016
- **Related, Embedded**
  When (and why) is it a good idea to use an FPGA in your embedded system design?
  2017
- **Moore, Andrew & Wilson, Ron**
  FPGAs for Dummies
  Intel
  2017

BIBLIOGRAPHIE 10

- **Staff, I. B. M. Research Editorial**
  IBM Scientifs Demonstrate In-memory Computing with 1 Million Devices for Applications in AI
  Blog, I. B. M. Research
  2017
- **Sebastian, Abu; Tuma, Tomas; Papandreou, Nikolaos; Gallo, Manuel Le; Kull, Lukas & Eleftheriou, Thomas Parnell \& Evangelos**
  Temporal correlation detection using computational phase-change memory
  communications, Nature
  2017
- **Sato, Kaz; Young, Cliff & Patterson, David**
  An in-depth look at Google's first Tensor Processing Unit (TPU)
  Google
  2017
- **Dean, Jeff & Hölzle, Urs**
  Build and train machine learning models on our new Google Cloud TPUs
  Google
  2017
- **Wood, Lamont**
  CPU architecture after Moore's law: what's next?
  Computerworld
  2017
- **Hardwarebee**
  Field Programmable Gate Array (FPGA) History and Applications
  2018

BIBLIOGRAPHIE 11

- **Reese, Lynnette**
  Comparing hardware for artificial intelligence: FPGAs vs. GPUs vs. ASICs
  Solutions, Embedded Intel
  2018
- **Arrow**
  FPGA vs CPU vs GPU vs Microcontroller
  2018
- **Dubuc, Damien**
  Afin de terminer notre série de billets, voici quelques réflexions et perspectives que nous ressortons de l'étude
  Aneo
  2018
- **Castells-Rufas, David**
  Workshop: programming FPGAs with OpenCL
  Cephis
  2018
- **Staff, I. B. M. Research Editorial**
  IBM Scientists Demonstrate Mixed-Precision In-Memory Computing for the First Time; Hybrid Design for AI Hardware
  Blog, I. B. M. Research
  2018
- **Thornton, Scott**
  What's the difference between Von-Neumann and Harvard architectures?
  Tips, Microcontroller
  2018

BIBLIOGRAPHIE 12

- **Arenas, Aaron**
  Introduction to FPGA design in Quartus
  Intel
  2018
- **Haff, Gordon**
  What comes after Moore's law
  Project, The Enterprisers
  2018
- **Gartenberg, Chaim**
  How Intel’s 9th Gen chips show the way forward after Moore’s Law
  Verge, The
  2018
- **Rayome, Alison DeNisco**
  How programming will change over the next 10 years: 5 predictions
  TechRepublic
  2018
- **Burt, Jeffrey**
  FPGA make Xilinx says the future of computing if ACAP
  Platform, The Next
  2018
- **Waalsdorp, Museum**
  Computers for electronic and mechanical engineering
  2019

BIBLIOGRAPHIE 13

- **Altera, Intel**
  Cyclone V Device Handbook: Volume 1: Device Interfaces and Integration, Logic Array Blocks and Adaptive Logic Modules in Cyclone V Devices
  2019
- **BitFusion**
  BitFusion, the elastic AI infrastructure for multi-cloud
  2019
- **ViPress.net**
  Infineon et NXP devant STMicroelectronics au 1er trimestre 2019
  2019
- **Altera, Intel**
  Cyclone V Device Datasheet
  2019
- **Cantrill, Bryan**
  No Moore Left to Give
  Joyent
  2019
- **Suryavansh, Manu**
  Google Coral Edge TPU Board Vs NVIDIA Jetson Nano Dev board - Hardware comparison
  Science, Towards Data
  2019
- **Sterckval, Sam**
  Google Coral Edge TPU vs NVIDIA Jetson Nano : A quick deep dive into Edge AI performance
  Noteworthy
  2019

BIBLIOGRAPHIE 14

- **Blog, Grus**

  Comparison of two new machine learning accelerators, Coral and Jetson Nano

  2019
- **Feldman, Michael**

  With Agilex, Intel gets a coherent FPGA strategy

  Platform, The Next

  2019