Introduction

Corinne FAY

01/10/2026

Rapport de validation

Produit : SOC Tracking

Table des matières

1. Introduction 5
2. Convertisseur Axi4s VGA 6
2.1. Validation du bloc - simulation 6
2.1.1. Test bench 6
2.1.2. Validation fonctionnelle 6
2.1.3. Démarrage après reset 7
2.1.4. Validation du timings hsync sur un transmission de ligne 8
2.1.5. Validation du timings vsync sur un transmission de frame 9
2.1.6. Validation des compteurs et signaux axis et des données 10
2.1.7. Résultat de test bench 11
3. Pattern generator 11
3.1. Validation du bloc pattern generator - simulation 11
3.1.1. Test bench 11
3.1.2. Validation fonctionnelle 11
3.1.3. Démarrage après reset 12
3.1.4. Zoom sur un changement de frame 12
3.1.5. Résultat de test bench 12
3.2. Validation du pattern generator – Intégration dans le top 13
3.2.1. Validation fonctionnelle 13
3.2.2. Démarrage après reset 14
3.2.3. Communication entre les pattern generator et l'axi4s_to_vga 15
3.2.4. Fin de ligne sur le pattern generator 16
3.2.5. 16
3.2.6. Validation du timings hsync sur un transmission de ligne 17
3.2.7. Validation du timings vsync sur un transmission de frame 17
3.3. Test sur carte 18
3.3.1. Synthèse et implémentation : rapport d'utilisation 18
3.3.2. Synthèse et implémentation : rapport de timings 18
3.3.3. Mesure ILA 19
3.3.1. Mesure sur oscilloscope 19
3.4. Démo 20
4. Le Filtre gris 21
4.1. Validation du bloc - simulation 21
4.1.1. Test bench 21
4.1.2. Validation fonctionnelle 21
4.1.3. Démarrage après reset et premier transfert de donnée 22
4.1.4. Changement de ligne , changement de frame 23

4.1.5. Calcul du filtre gris 24
4.1. Validation du filtre gris – Intégration dans le top 25
4.1.1. Validation fonctionnelle 25
4.1.2. Démarrage après reset 26
4.1.3. Communication axi stream entre les différents blocs 27
4.1.4. Changement de ligne , ...28
4.1.5. Validation des timings hsync 28
4.1.6. Validation du timings vsync sur un transmission de frame 29
4.1. Test sur carte 29
4.1.1. Synthèse et implémentation : rapport d'utilisation 29
4.1.1. Synthèse et implémentation : rapport de timings 29
4.1.1. Mesure sur oscilloscope 30
4.1. Démo 31
5. Le Filtre de gauss 32
5.1. Validation du bloc - simulation 32
5.1.1. Test bench 32
5.1.2. Validation fonctionnelle 32
5.1.3. Démarrage après reset et premier transfert de données 33
5.1.1. Calcul du filtrage : line buffer et matrice 34
5.1.2. Calcul du filtre gauss 35
5.1. Validation du filtre de gauss – Intégration dans le top 36
5.1.1. Validation fonctionnelle 36
5.1.2. Démarrage après reset 37
5.1.3. Communication axi stream autour du filtre de gauss 38
5.1.4. Validation des timings hsync 39
5.1.5. Validation du timings vsync sur un transmission de frame 39
5.1. Test sur carte 40
5.1.1. Synthèse et implémentation : rapport d'utilisation 40
5.1.1. Synthèse et implémentation : rapport de timings 40
5.1.2. Mesure ILA 41
5.1.3. Mesure sur oscilloscope 41
5.1. Démo 41
6. Le Filtre harris 43
6.1. Validation du bloc - simulation 43
6.1.1. Test bench 43
6.1.2. Validation fonctionnelle 43
6.1.3. Démarrage après reset et premier transfert de données 44
6.1.4. Calcul du filtrage : line buffer et matrice 45

Introduction
6.1.5. Calcul du filtre harris 45
6.2. Validation du filtre de gauss – Intégration dans le top 47
6.2.1. Validation fonctionnelle 47
6.2.2. Démarrage après reset 48
6.2.3. Communication axi stream autour du filtre harris 49
6.2.4. Validation des timings hsync 50
6.2.5. Validation du timings vsync sur un transmission de frame 50
6.3. Test sur carte 51
6.3.1. Synthèse et implémentation : rapport d'utilisation 51
6.3.2. Synthèse et implémentation : rapport de timings 51
6.3.3. Mesure ILA 52
6.3.4. Mesure sur oscilloscope 52
6.4. Démo 53
7. Ligne complète 54
7.1. Validation de la ligne complète en simulation 54
7.1.1. Protocole axi stream sur la chaine 54
7.1.2. Validation des timings hsync 55
7.1.3. Validation du timings vsync sur un transmission de frame 55
7.1.4. Validation d'un overlay 56
7.2. Test sur carte 56
7.2.1. Synthèse et implémentation : rapport d'utilisation 56
7.2.2. Synthèse et implémentation : rapport de timings 57
7.2.3. Mesure ILA 57
7.2.4. Mesure sur oscilloscope 58
7.3. Démo 59
8. Architecture complète 60
9. Conclusion 63

1. Introduction

Ce rapport de validation présente les simulation et mesures faites conformément au plan de validation.
L'objectif est d'afficher sur écran VGA une image fournie par un fichier stocké en mémoire DDR ou fournie par un pattern generator.
Cette image pourra être affichée directement sans filtrage ou avec superposition d'un filtrage
Le filtrage est composé de plusieurs étages : filtre gris, filtre de gauss filtre harris et seuillage . Le but est d'identifier les contours de l'image
Ce rapport de validation Va présenter les simulations faites sur les blocs unitairement puis sur la chaine de filtrage puis les mesures et observation sur carte.

La validation va se dérouler dans l'ordre ci-dessous afin de pouvoir valider les blocs indépendamment puis intégrés dans le système.

- Convertisseur axi4s ti VGA
- Pattern generator
- Filtre gris
- Filtre de Gauss
- Filtre Harris
- Doubleurs, fifo et overlay
- DMA
- Système complet

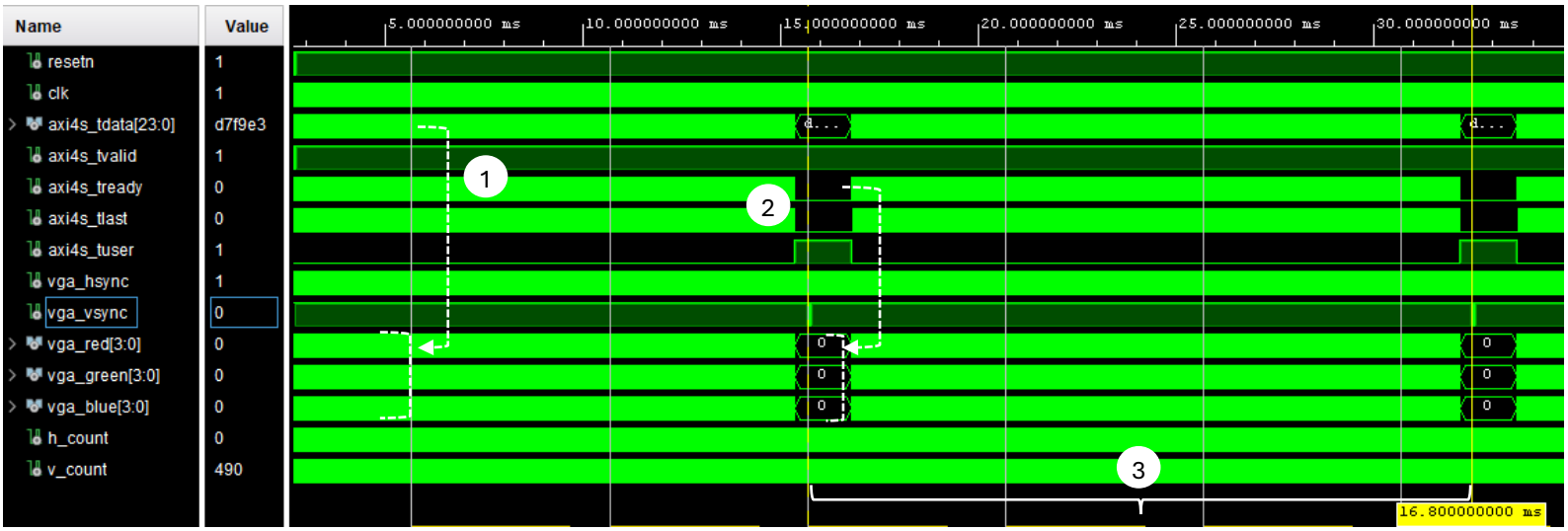
2. Convertisseur Axi4s VGA

2.1. Validation du bloc - simulation

2.1.1. Test bench

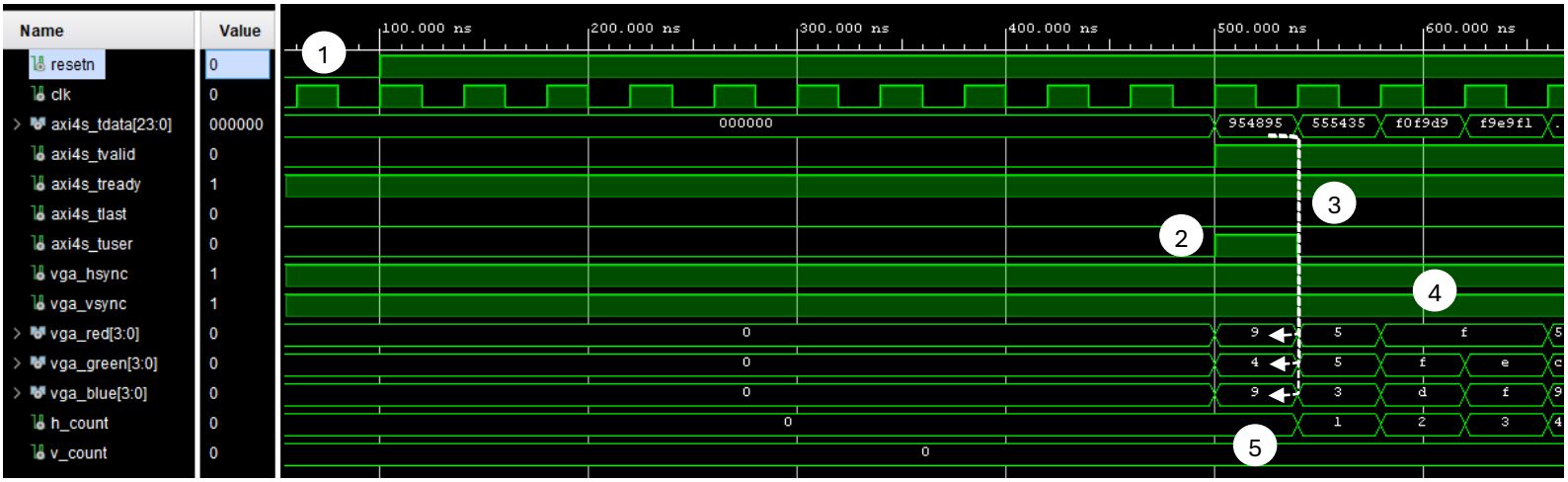
Le test bench utilisé consiste à utiliser le générateur de trame axis4s fournit (axi4sdriver. vhd)
axi4sdriver fournit une trame axi4s à partir d'un fichier texte en entrée.
Le fichier texte doit contenir un table de 640*480 mots de 8bit en décimal
Afin de faciliter la validation du transfert des données, le fichier texte utilisé en entrée a été rempli de mots de 8 bits générés de façon aléatoire et de sorte que 2 nombres consécutifs ne soient pas identiques

2.1.2. Validation fonctionnelle



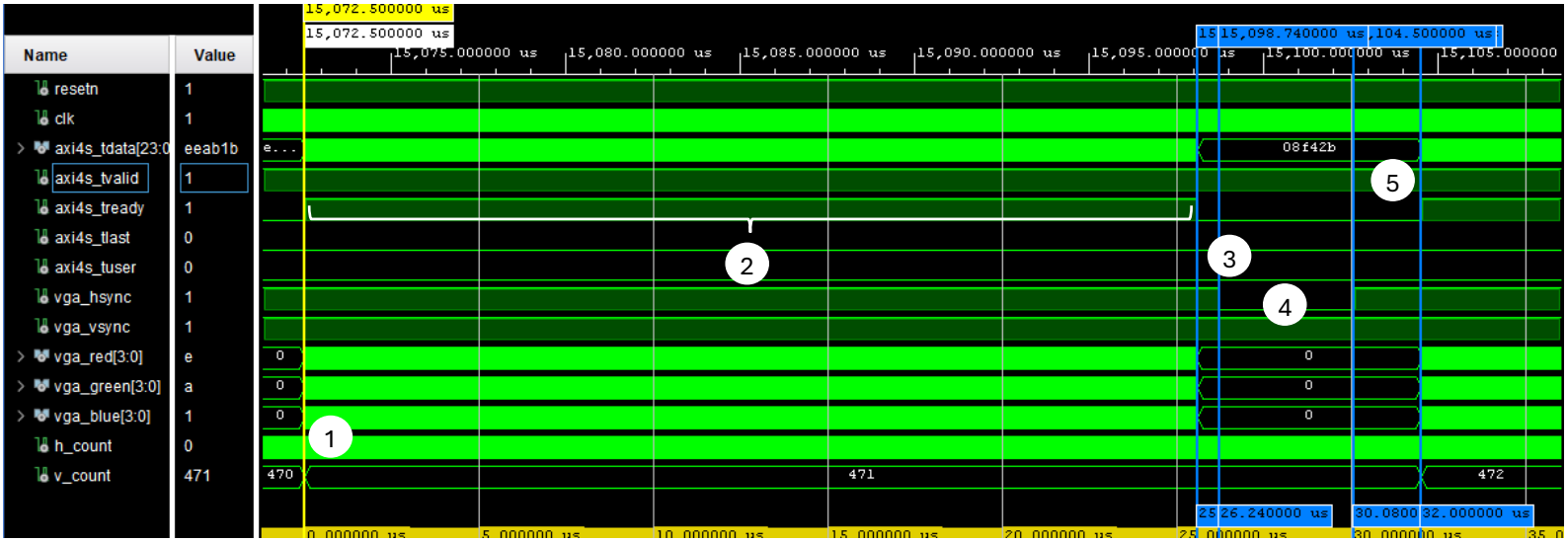
1. Les données tdata sont transmises sur les signaux red,green,blue du VGA quand le handshake (tvalid & tready) a lieu.
2. La transmission des données est arrêtée dès que un des signaux du handshake passe à 0 . Ici le tready = 0 bloque la transmission
3. La durée d'une frame VGA entre 3 vsync dure16.8ms soit 800*525cycle de 25MHz

2.1.3. Démarrage après reset



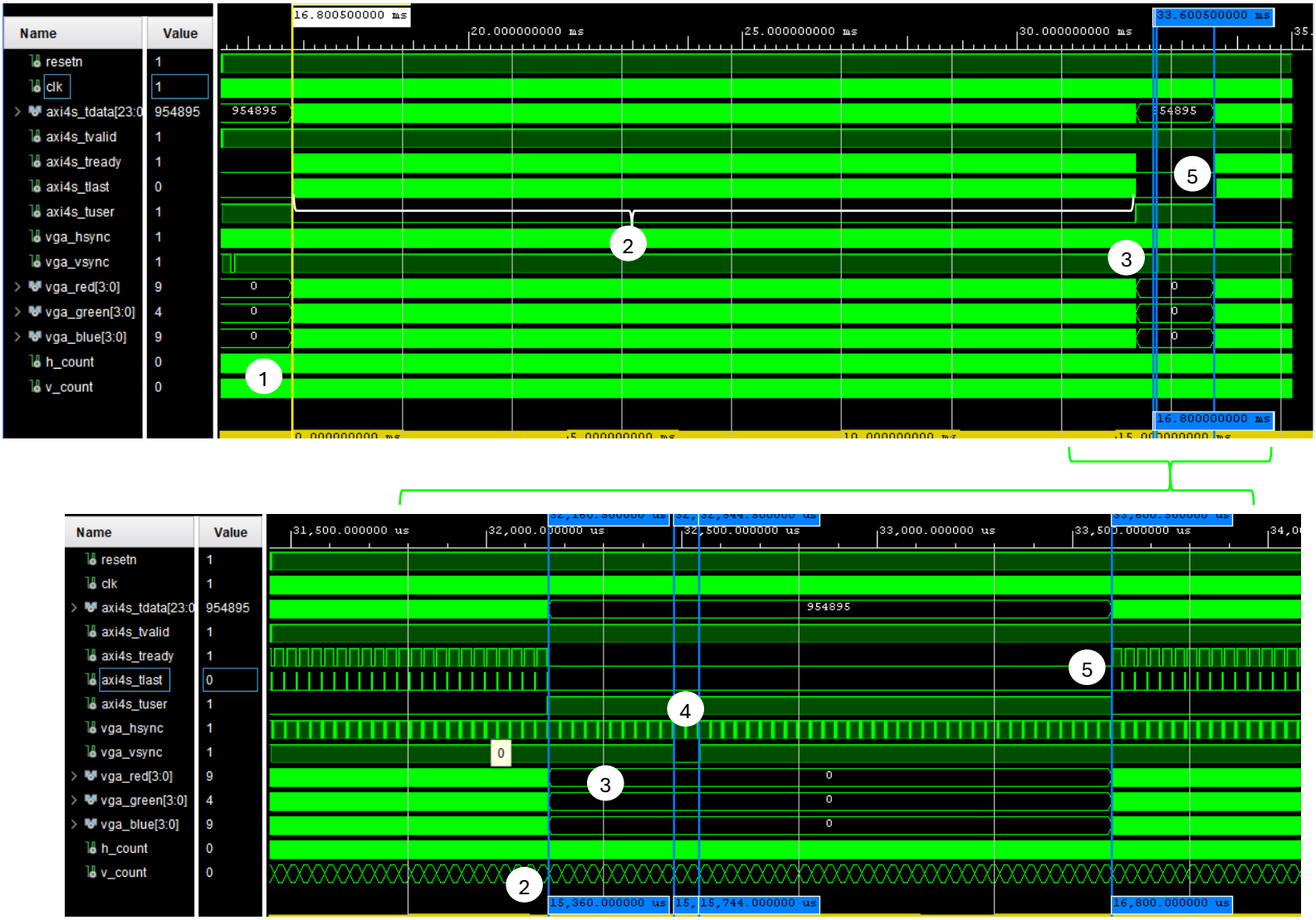
1. Au reset: les signaux red, green, blue VGA sont à 0
   les signaux de contrôle VGA sont à 1
   les compteurs VGA sont à 0
2. La trame VGA commence après son délai d'initialisation de 10*40ns
   tuser passe à'1' sur le premier pixel transmis
   tvalid passe à 1 : les données sont transmisses
3. Le premier pixel est transmis p(0,0) =0x954895
   les signal rgb du VGA prend les valeurs MSB de chaque couleur codées sur 8*24 bit sur tdata
   red = 9 = tdata(23 to 20)
   green = 4 = tdata(15 to 12)
   blue = 9 = tdata(7 to 4)
4. Les données suivantes sont transmises tant que le handshake est ok
5. les compteurs VGA s'incrément dès le démarrage de la transmission à chaque transmission

2.1.4. Validation du timings hsync sur un transmission de ligne



1. Au changement de ligne v_count s'incrémente et h_count est remis à zéro
   les signaux de contrôle VGA sont à 1
   les compteurs VGA sont à 0
2. Les données sont transmises tant que tready = 1 le handshake est ok
   La transmission s'arrêtes quand h_count = 639
   La transmission dure 25.6 us = 640*periode(25Mhz)
3. Le signal hsync passe à 0 16 periode plus tard à t= 26.24us du début de transmission
   26.24 us = (640+16)* periode(25Mhz)
4. Le signal hsync reste à 0 pendant 96 periode , il repasse à 1 à t= 30.08us
   30.08us = (640+16+96)* periode(25Mhz)
5. Le protocole VGA dis d'attendre 48 periode supplémentaires avant de lancer la transmission d'un autre ligne
   Le signal tready repasse à 1 à t= 32us pour relancer la transmission d'un autre mot
   32us = (640+16+96+48)* periode(25Mhz)

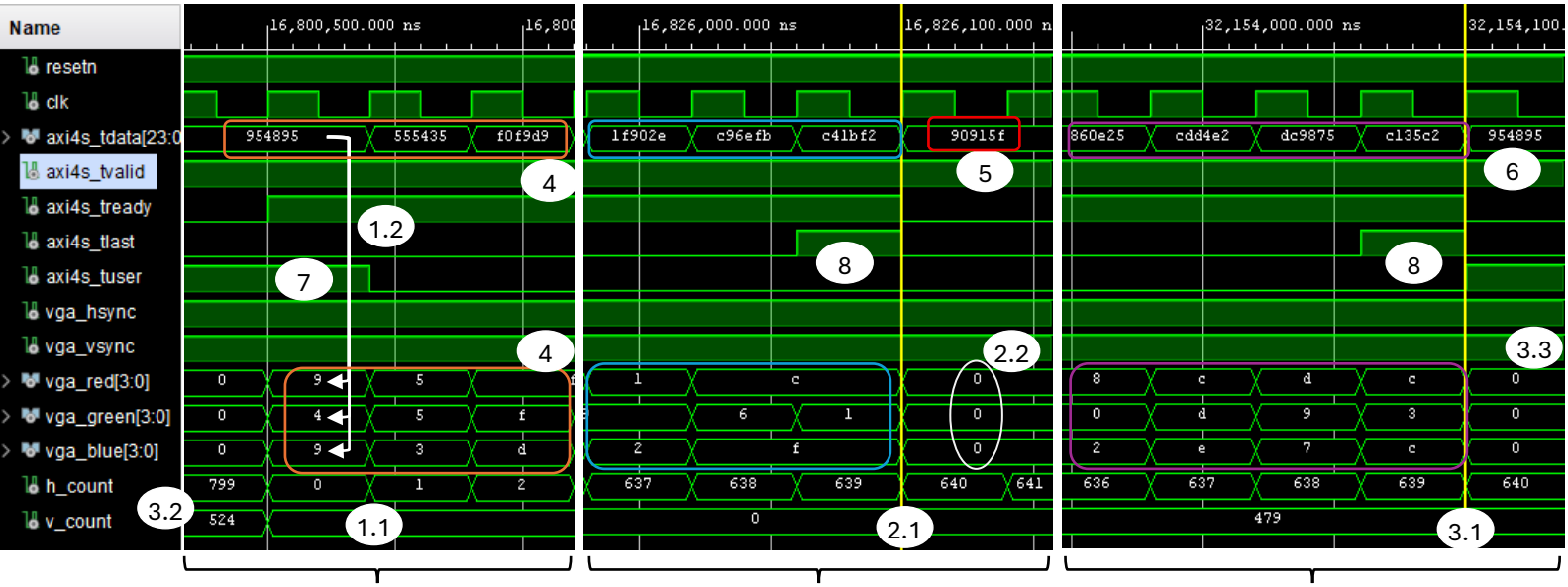
2.1.5. Validation du timings vsync sur un transmission de frame



1. Au changement de frame v_count et h_count sont remis à zéro
   Les données commencent à être transférées en commençant par la première ligne (v_count = 0)
2. Le transfert des 480 lignes dure 16.8ms = 800*480*periode(25Mhz)
3. Le signal vsync attend 10 lignes avant de passer à 0 à t = 15.68ms = 800*(480+10)*periode(25Mhz)
4. Le signal vsync reste à 0 pendant 2 lignes, il repasse à 1 à t = 15.744ms = 800*(480+10+2)*periode(25Mhz)
5. La transmission est relancée quand l'équivalent de 33 cycles de lignes sont passés à t=16.8ms = (800*(480+10+2+33))*periode(25Mhz)

Convertisseur Axi4s VGA

2.1.6. Validation des compteurs et signaux axis et des données



4

1 Début de frame et de ligne

image_alea_r_hexa.txt

1	95	55	F0	F9	51	3E	F6
2	90	E5	D5	67	80	A2	17
3	D2	45	20	11	31	3C	1
4	DB	44	EC	E	32	F3	9E

image_alea_g_hexa.txt

1	48	54	F9	E9	CF	40	3
2	91	6A	7C	C9	6E	31	1C
3	25	62	75	FC	5D	59	1B
4	62	4F	D8	E4	24	E3	C0

image_alea_b_hexa.txt

1	95	35	D9	F1	94	E9	D0
2	5F	72	6A	DA	86	F	2B
3	24	19	5B	2C	4	54	77
4	FD	6E	57	1	44	A4	65

2 Fin de première ligne

1	81	1C	20	6A	1F	C9	C4
2	7F	AD	23	D4	5E	75	5B
3	90	7E	8	8F	DD	5F	91
4	6E	53	30	D6	32	D1	DC

1	64	2F	41	9A	90	6E	1B
2	44	A0	86	3B	70	A9	63
3	9C	B7	DC	91	6D	E7	55
4	25	50	55	D5	97	C3	24

1	92	59	E	30	2E	FB	F2
2	20	32	18	33	30	AF	3A
3	1E	30	C1	D4	70	DE	D
4	57	96	E2	B1	3F	A8	C4

3 fin de frame et fin de ligne

477	8B	4A	5	9B	F3	2B	E7
478	6	8E	C0	A8	1F	5A	58
479	69	81	91	C2	F8	A5	D0
480	F1	A8	AA	86	CD	DC	C1

477	92	F6	F9	35	7C	D6	DC
478	B	21	D7	B0	36	7F	1E
479	FC	F5	70	BE	6C	91	E7
480	20	32	A1	E	D4	98	35

477	59	BE	41	79	EA	64	3D
478	88	8F	9B	93	42	CD	DB
479	89	EB	C2	A4	CA	C4	7B
480	4D	D3	3B	25	E2	75	C2

1. 1.1Début de frame : v_count est remis à zéro
   Début de ligne ; h_count est remis à zéro
   1.2La donnée est transmise quand le handshake est OK
2. 2.1Fin de ligne : h_count =639 sur le dernier pixel transmis
   Le compteur h_count ne sera remis à zéro qu'à la fin du protocole VGA
   2.2La donnée n'est plus transmise elle est misé à zéro
3. 3.1Fin de frame : v_count = 479
   Fin de ligne : h_count =639 sur le dernier pixel transmis
   3.2Les compteurs h_count et v_count ne seront remis à zéro qu'à la fin du protocole VGA et au début de la nouvelle frame
   3.3La donnée n'est plus transmise elle est misé à zéro
4. Vérification des données : on compare les données des fichiers texte d'entrées pour le chargement des 3 couleurs avec les valeurs dans tdata et dans vga_rgb
   Dans tdata nous avons bien la concaténation des valeurs rgb des trois fichiers,
   Les MSB de ces valeurs se retrouvent bien sur les signaux rgb du VGA
5. Sur une fin de ligne la prochaine donnée attend le handshake sur tdata . Ici c'est le premier pixel de la seconde ligne
6. Sur une fin de frame la prochaine donnée attend le handshake sur tdata . Ici c'est le premier pixel de la première ligne

Pattern generator

7. Le signal tuser est à un sur le premier pixel de la frame
8. Le signal tlast passe à 1 sur le dernier pixel de chaque ligne

2.1.7. Résultat de test bench

```
# run 80000us
Note: 480 lignes de 640pixels dans la frame transmises
Time: 16800540 ns  Iteration: 1  Process: /tb_axi4s_to_vga/count  File: D:/Vivado/PROJET/axi4s_to_vga/src/tb_axi4s_to_vga.vhd
Note: 480 lignes de 640pixels dans la frame transmises
Time: 33600540 ns  Iteration: 1  Process: /tb_axi4s_to_vga/count  File: D:/Vivado/PROJET/axi4s_to_vga/src/tb_axi4s_to_vga.vhd
Note: 480 lignes de 640pixels dans la frame transmises
Time: 50400540 ns  Iteration: 1  Process: /tb_axi4s_to_vga/count  File: D:/Vivado/PROJET/axi4s_to_vga/src/tb_axi4s_to_vga.vhd
Note: 480 lignes de 640pixels dans la frame transmises
Time: 67200540 ns  Iteration: 1  Process: /tb_axi4s_to_vga/count  File: D:/Vivado/PROJET/axi4s_to_vga/src/tb_axi4s_to_vga.vhd
run: Time (s): cpu = 00:00:11 ; elapsed = 00:00:13 . Memory (MB): peak = 1425.492 ; gain = 0.000
xsim: Time (s): cpu = 00:00:13 ; elapsed = 00:00:15 . Memory (MB): peak = 1425.492 ; gain = 0.000
INFO: [USF-XSim-96] XSim completed. Design snapshot 'tb_axi4s_to_vga_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 80000us
launch_simulation: Time (s): cpu = 00:00:13 ; elapsed = 00:00:20 . Memory (MB): peak = 1425.492 ; gain = 0.000
```

Le test bench confirme que le nombre de pixels et de lignes est bien transmis à chaque frame.

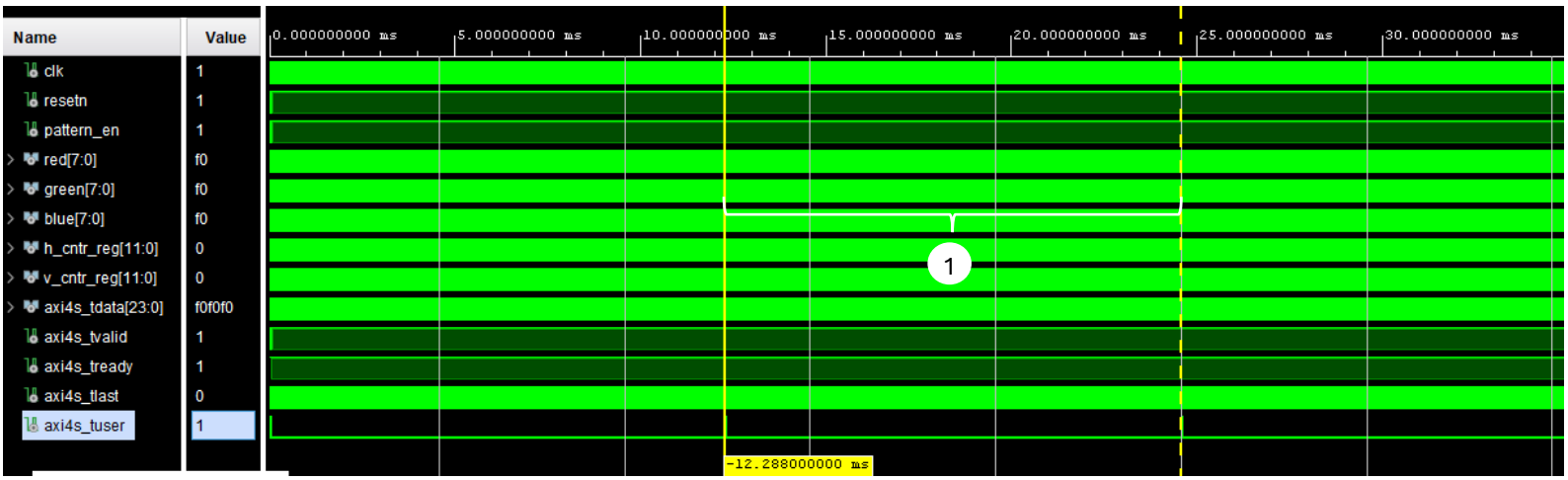
3. Pattern generator

3.1. Validation du bloc pattern generator - simulation

3.1.1. Test bench

Le test bench utilisé consiste démarrer le pattern sur pattern_en et autoriser ou non la transaction sur tready

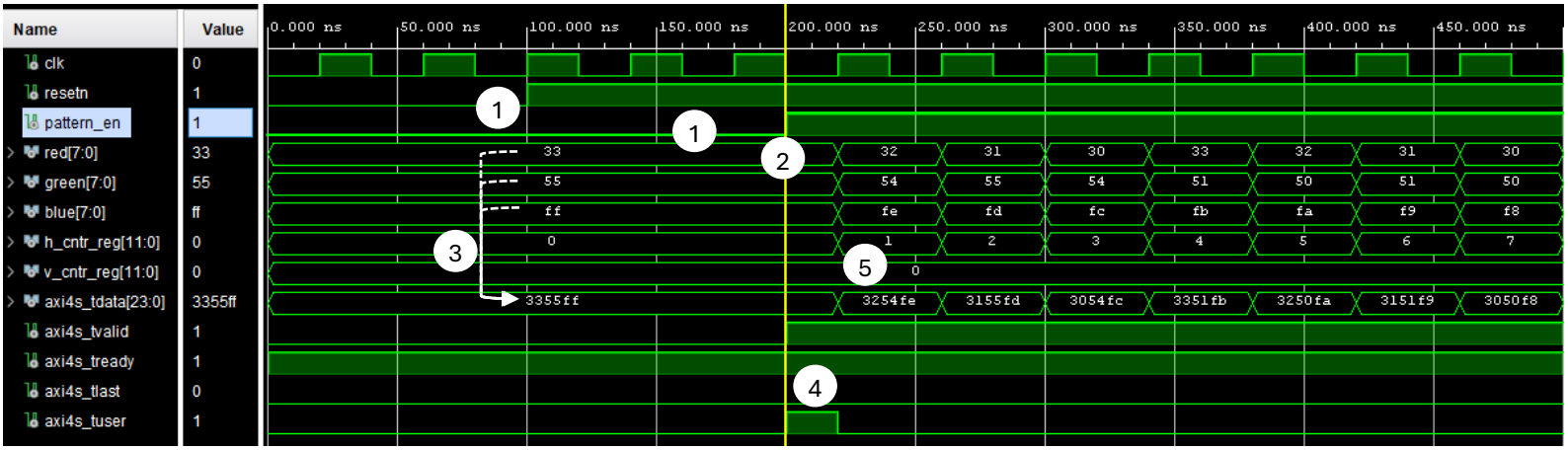
3.1.2. Validation fonctionnelle



1. Le pattern generator crée trois signaux red, green, blue qui sont transmis sur l'axi stream.
   Chaque frame est délimitée par le signal tuser .
   Une frame est émise toutes les 12.288ms = 640*480*periode(25Mhz)

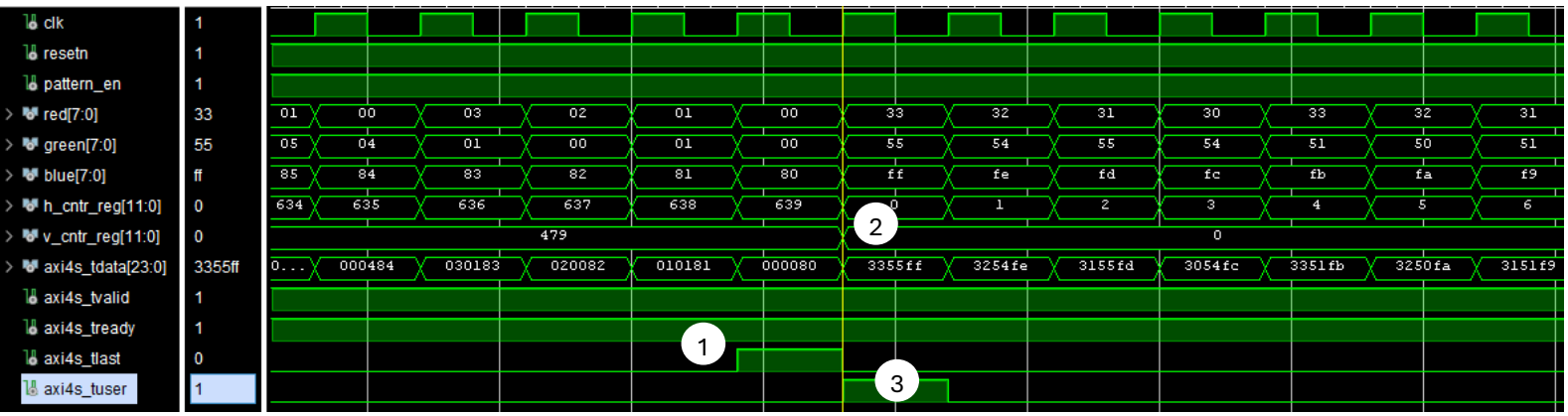
Pattern generator

3.1.3. Démarrage après reset



1. Au reset, le signaux de sortie de contrôle axi stream sont à zéro: tvalid, tuser, tlast.
   Le signal tready est à 1 pour indiquer que l'axi slave est prêt pour recevoir de la donnée.
   Les données sont générées mais ne sont pas envoyées
   Les compteur de frame sont à 0.
2. Le pattern generator est mis en marche pattern_en = 1
   Le tvalid suit pattern_en et indique que les données sont prêtes à être envoyées
3. tdata est généré en concaténant les trois couleurs générées par le pattern.
4. tuser passe à 1 sur le premier pixel de la frame
5. le compteur h_count s'incrémente sur chaque pixel transmis

3.1.4. Zoom sur un changement de frame



1. tlast passe à 1 sur le dernier pixel de la ligne
2. Les compteurs sont remis à 0
3. tuser passe à 1 sur le premier pixel de la ligne

3.1.5. Résultat de test bench

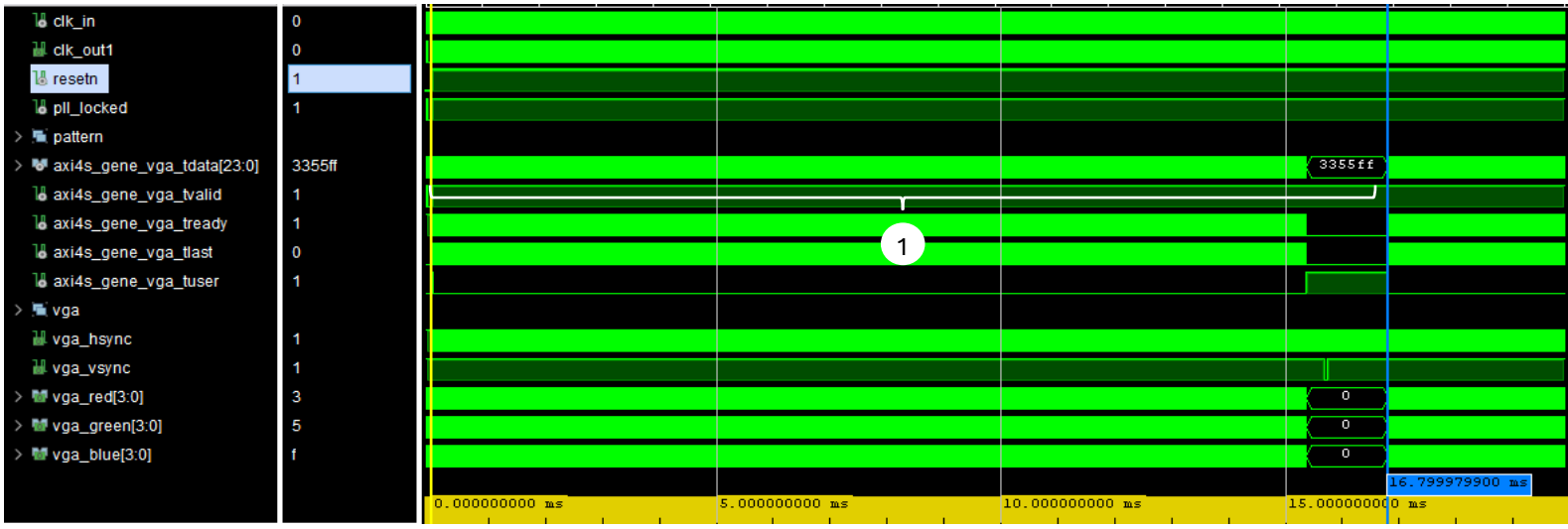
```
# run 35000us
Note: 480 lignes de 640pixels dans la frame transmises
Time: 12288220 ns  Iteration: 1  Process: /tb_pattern_generator/count  File: D:/Vivado/PROJET/pattern_generator/src/tb_pattern_generator.vhd
Note: 480 lignes de 640pixels dans la frame transmises
Time: 24576220 ns  Iteration: 1  Process: /tb_pattern_generator/count  File: D:/Vivado/PROJET/pattern_generator/src/tb_pattern_generator.vhd
run: Time (s): cpu = 00:00:06 ; elapsed = 00:00:06 . Memory (MB): peak = 1318.965 ; gain = 0.000
xsim: Time (s): cpu = 00:00:07 ; elapsed = 00:00:08 . Memory (MB): peak = 1318.965 ; gain = 0.000
INFO: [USF-XSim-96] XSim completed. Design snapshot 'tb_pattern_generator_behav' loaded.
```

Le test bench confirme que le nombre de pixels et de lignes est bien transmis à chaque frame.

3.2. Validation du pattern generator – Intégration dans le top

Les blocs présent pour cette simulation sont la pll, le pattern generator et le convertisseur axi4s_to_vga

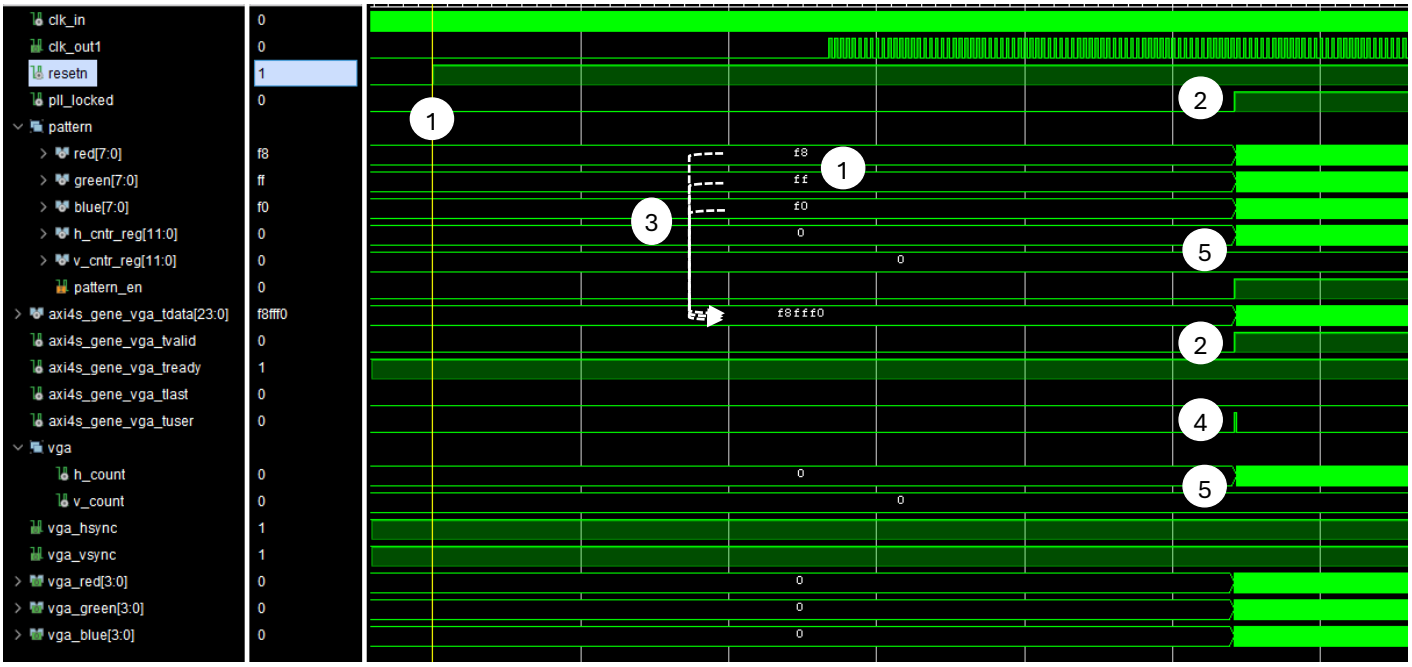
3.2.1. Validation fonctionnelle



1. Le pattern generator crée une frame axi stream contentant une image qui est transmise sur le convertisseur axi4s_to_vga
2. Cette communication respecte le protocole VGA
3. Une frame est émise toutes les 16.8ms = 800*525*periode(25Mhz)
   Chaque frame est délimitée par le signal tuser.

Pattern generator

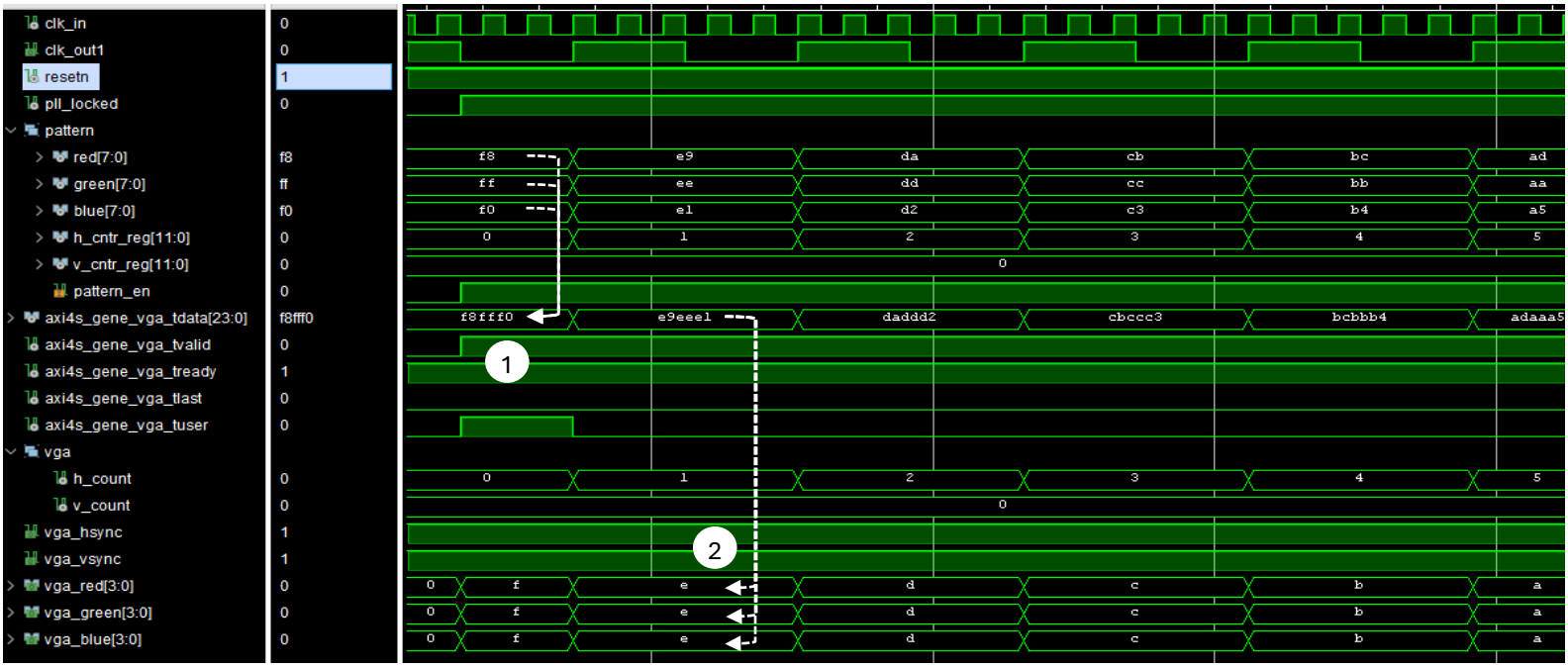
3.2.2. Démarrage après reset



1. Au reset, le signaux de sortie de contrôle axi stream sont à zéro: tvalid, tuser, tlast.
   Le signal tready est à 1 pour indiquer que l'axi slave est prêt pour recevoir de la donnée.
   Les données sont générées mais ne sont pas envoyées
   Les compteur de frame sont à 0.
2. Le pattern generator est mis en marche quand la pll a démarré sur pll_locked
   Le tvalid suit pattern_en et indique que les données sont prêtes à être envoyées
3. tdata est généré en concaténant les trois couleurs générées par le pattern.
4. tuser passe à 1 sur le premier pixel de la frame
5. les compteurs du pattern generator et du vga démarrent, notamment les compteurs de pixel , les compteurs de lignes s'incrémenterons plus tard

Pattern generator

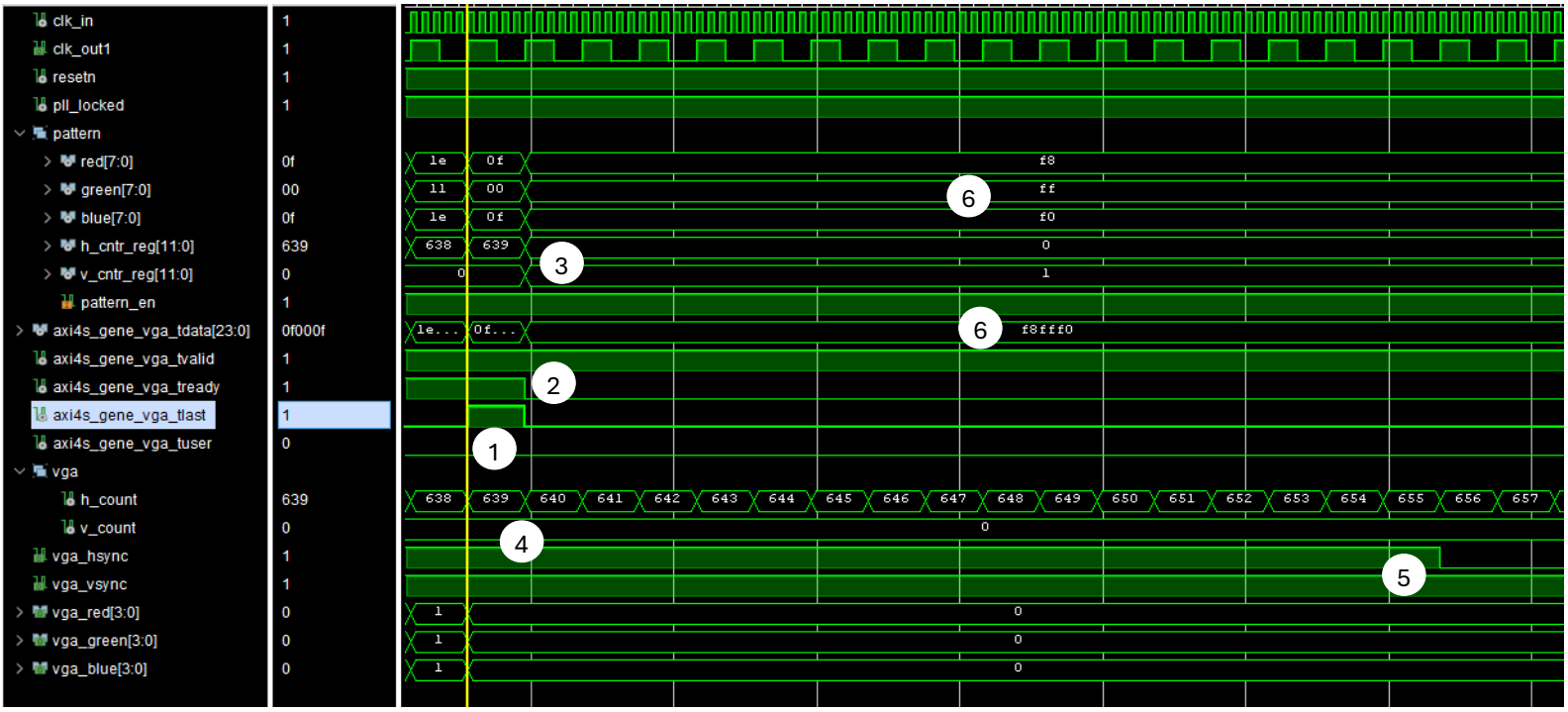
3.2.3. Communication entre les pattern generator et l'axi4s_to_vga



1. Le premier pixel est envoyé dès que le handshake est OK:
   axi4s_gene_vga_tvalid = 1 et axi4s_gene_vga_tready = 1
2. Les donnée sont transmises de axi4s_gene_vga_tdata vers les trois signaux de couleur du vga en gardant uniquement les MSB

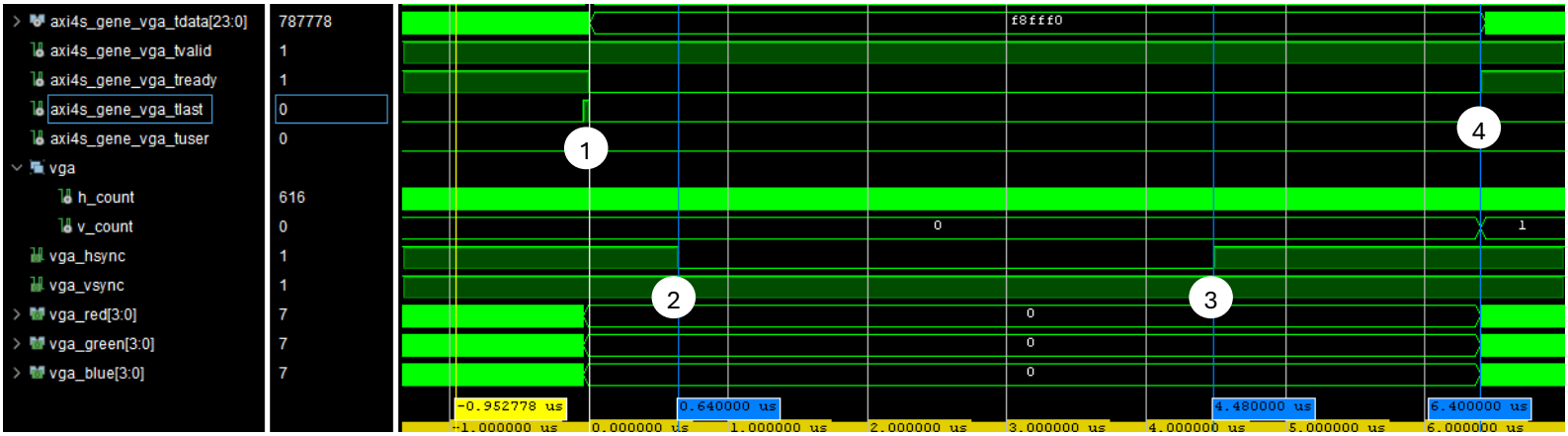
Pattern generator

3.2.4. Fin de ligne sur le pattern generator



1. Fin de ligne sur le pattern generator : axi4s_gene_vga_tlast = 1 , h_cntr_reg = 639
   La dernière data est transmise au VGA (Dans notre cas , elle vaut 0x0f000f soit tous les signaux du vga à zéro)
   axi4s_gene_vga_tvalid = 1 et axi4s_gene_vga_tready = 1
2. Le signal tready est mis à zéro par le axi4s-to_vga : plus de transfert autorisé , on va arriver dans le hsync du vga
3. Coté pattern generator : Le compteur de pixel est remis à zéro, le compteur de ligne s'incrémente
4. Coté axi4s-to_vga : Le compteur de pixel va continuer à compter jusqu'à 799
5. Hsync passe à 0 quand h_count = 639 +16
6. Coté pattern generator : La génération des données est arrêtée , le bus de data est bloqué sur lap prochaine donnée à transférer

3.2.5.

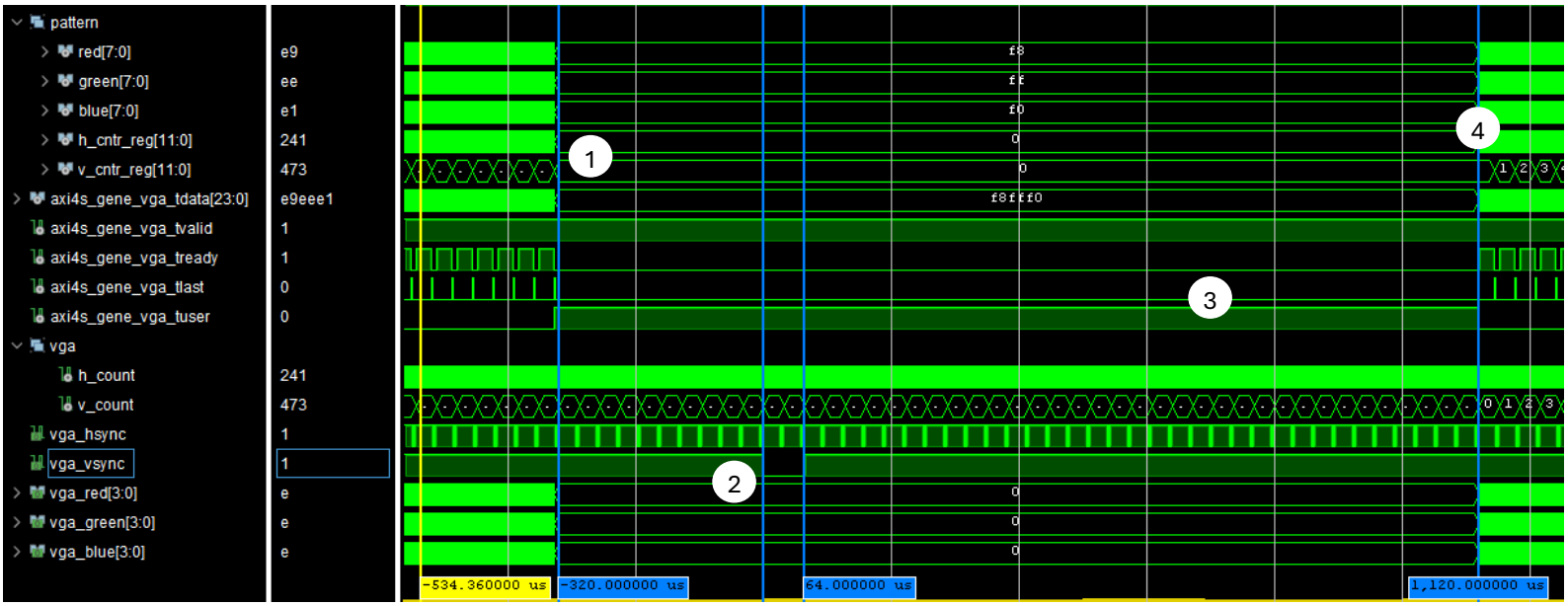


Pattern generator

3.2.6. Validation du timings hsync sur un transmission de ligne

1. Fin de ligne sur le pattern generator : axi4s_gene_vga_tlast = 1 , h_cntr_reg = 639
2. Le signal hsync passe à 0 16 periode plus tard à t=0.64us = 16* periode(25Mhz)
3. Le signal hsync reste à 0 pendant 96 periode , il repasse à 1 à 4.48us après la fin de ligne
   4.48us = (16+96)* periode(25Mhz)
4. Le protocole VGA dis d'attendre 48 periode supplémentaires avant de lancer la transmission d'un autre ligne
   Le signal tready repasse à 1 à t=6.4us après la fin de ligne pour relancer la transmission d'un autre mot. 6.4us = (16+96+48)* periode(25Mhz)

3.2.7. Validation du timings vsync sur un transmission de frame



1. Au changement de frame v_count et h_count sont remis à zéro
   Les données commencent à être transférées en commençant par la première ligne (v_count = 0)
2. Le signal vsync attend 10 lignes avant de passer à 0 soit 320us = 800*10*periode(25Mhz)après la fin de frame
3. Le signal vsync reste à 0 pendant 2 lignes, soit 64us = 800*2*periode(25Mhz)
4. La transmission est relancée quand 33 cycles de lignes sont passés à t= 1.120us
   (800*(2+33))*periode(25Mhz)

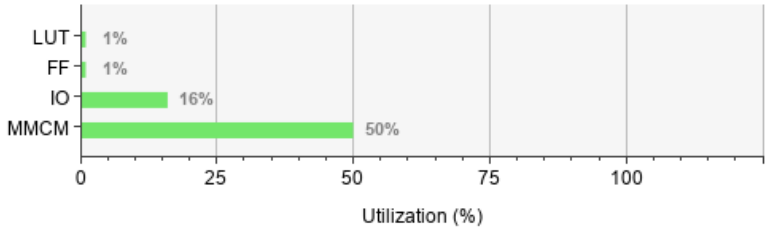
3.3. Test sur carte

3.3.1. Synthèse et implémentation : rapport d'utilisation

Name	Slice LUTs (17600)	Slice Registers (35200)	Bonded IOB (100)	BUFGCTRL (32)	MMCME2_ADV (2)
top_tpg_vga	186	96	16	2	1
u_axi4s_to_vga (axi4s_to_vga)	42	21	0	0	0
u_clk_wiz_0 (clk_wiz_0)	0	0	0	2	1
u_pattern_generator (pattern_generator)	143	75	0	0	0

Ref Name	Used	Functional Category
FDCE	90	Flop & Latch
LUT2	87	LUT
LUT4	53	LUT
LUT6	49	LUT
CARRY4	34	CarryLogic
LUT1	28	LUT
LUT5	23	LUT
OBUF	14	IO
LUT3	9	LUT
FDPE	6	Flop & Latch
IBUF	2	IO
BUFG	2	Clock
MMCME2_ADV	1	Clock

Resource	Utilization	Available	Utilization %
LUT	186	17600	1.06
FF	96	35200	0.27
IO	16	100	16.00
MMCM	1	2	50.00



Pour l'implémentation du axi4s_to_vga :

21 registres correspondent aux compteur de ligne et de colonne.

42 LUT pour la logique des comparateurs principalement

Pour l'implémentation du pattern generator :

75 registres correspondent aux compteur de ligne et de colonne, compteur sur les box du pattern

143 LUT pour la logique de construction du pattern principalemnet

3.3.2. Synthèse et implémentation : rapport de timings

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	33,837 ns	Worst Hold Slack (WHS):	0,062 ns	Worst Pulse Width Slack (WPWS):	2,000 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	164	Total Number of Endpoints:	164	Total Number of Endpoints:	102

All user specified timing constraints are met.

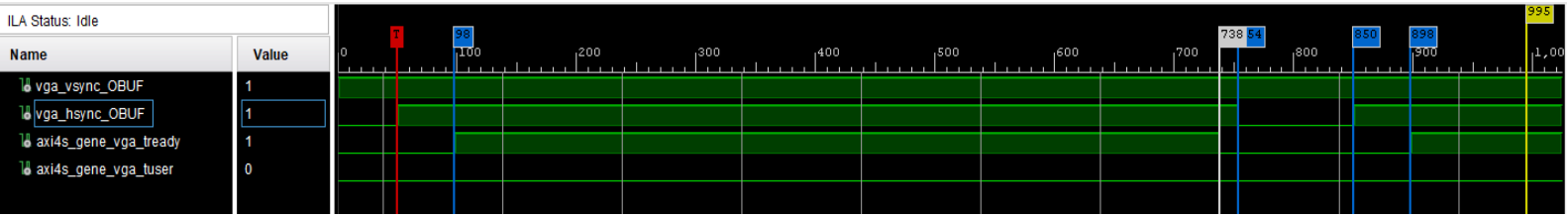
Les contraintes de timing sont respectées avec une horloge à 25Mhz.

Summary	
Name	Path 1
Slack	33.837ns
Source	u_pattern_generator/h_cntr_reg_reg[9]/C (rising edge-triggered cell FDCE clocked by clk_out1_clk_wiz_0 {rise@0.000ns fall@20.000ns period=40.000ns})
Destination	u_axi4s_to_vga/v_count_reg[0]/CE (rising edge-triggered cell FDCE clocked by clk_out1_clk_wiz_0 {rise@0.000ns fall@20.000ns period=40.000ns})
Path Group	clk_out1_clk_wiz_0
Path Type	Setup (Max at Slow Process Corner)
Requirement	40.000ns (clk_out1_clk_wiz_0 rise@40.000ns - clk_out1_clk_wiz_0 rise@0.000ns)
Data Path Delay	5.740ns (logic 1.275ns (22.212%) route 4.465ns (77.788%))
Logic Levels	5 (LUT3=1 LUT4=1 LUT5=1 LUT6=2)
Clock Path Skew	-0.057ns
Clock Un...rtainty	0.160ns

Le chemin critique se trouva dans la logique autour des compteur du pattern generator et du convertisseur axis vga

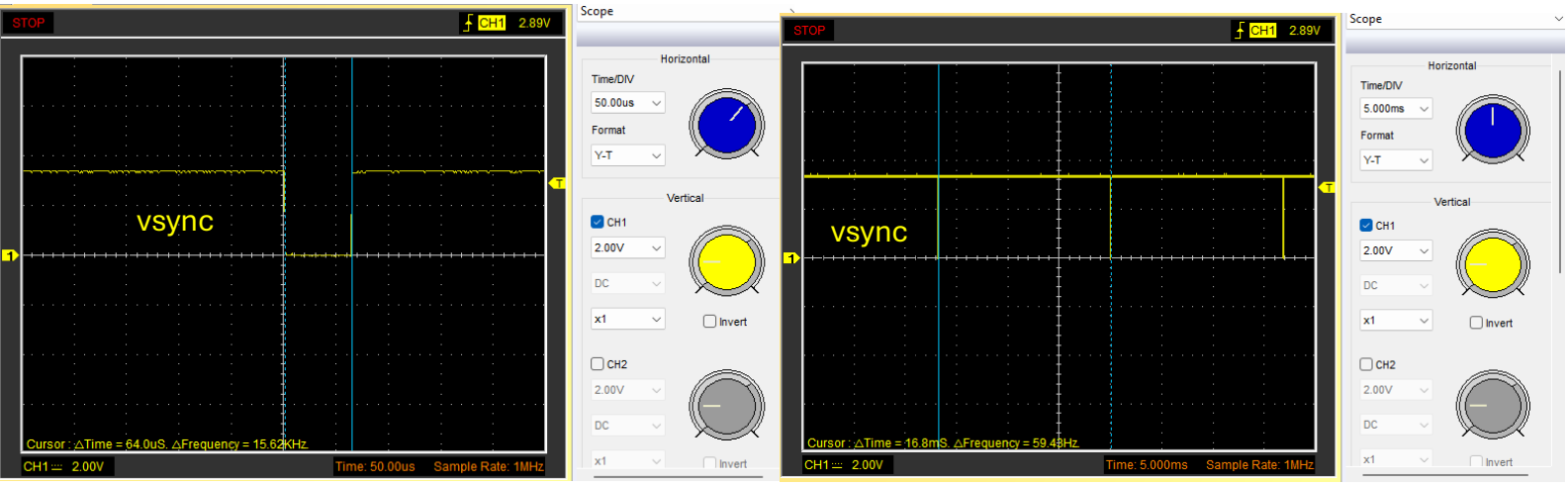
Pattern generator

3.3.3. Mesure ILA

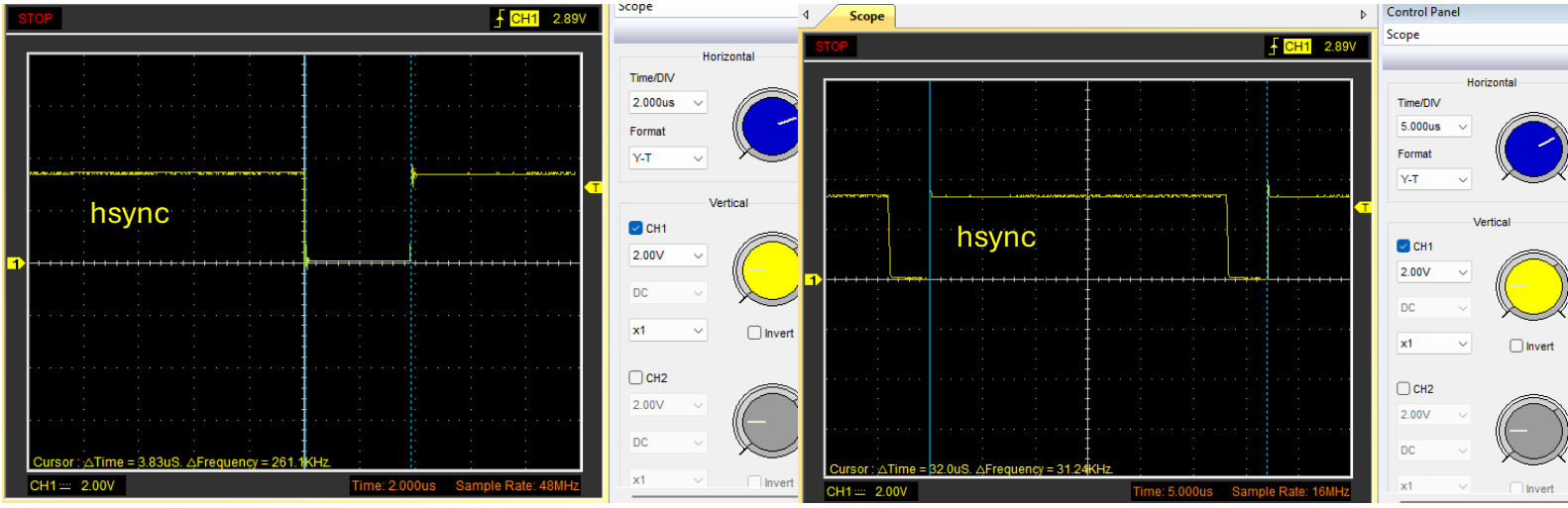


tready reste à 1 sur un ligne pendant 738-98= 640*periode(clk)
hsync attend 754-738 = 16*periode(clk)avant de passer à 0
hsync est à 0 pendant 850-754 = 96*periode(clk)
tready passe à 1 898-850 = 48*periode(clk) après que hsync soit repassée à 1

3.3.1. Mesure sur oscilloscope



Le signal vsync passe à 0 toutes les 16.8ms = 800*525* periode(25Mhz)
le signal vsync reste à 0 pendant 64us= 800*2*periode(25Mhz)

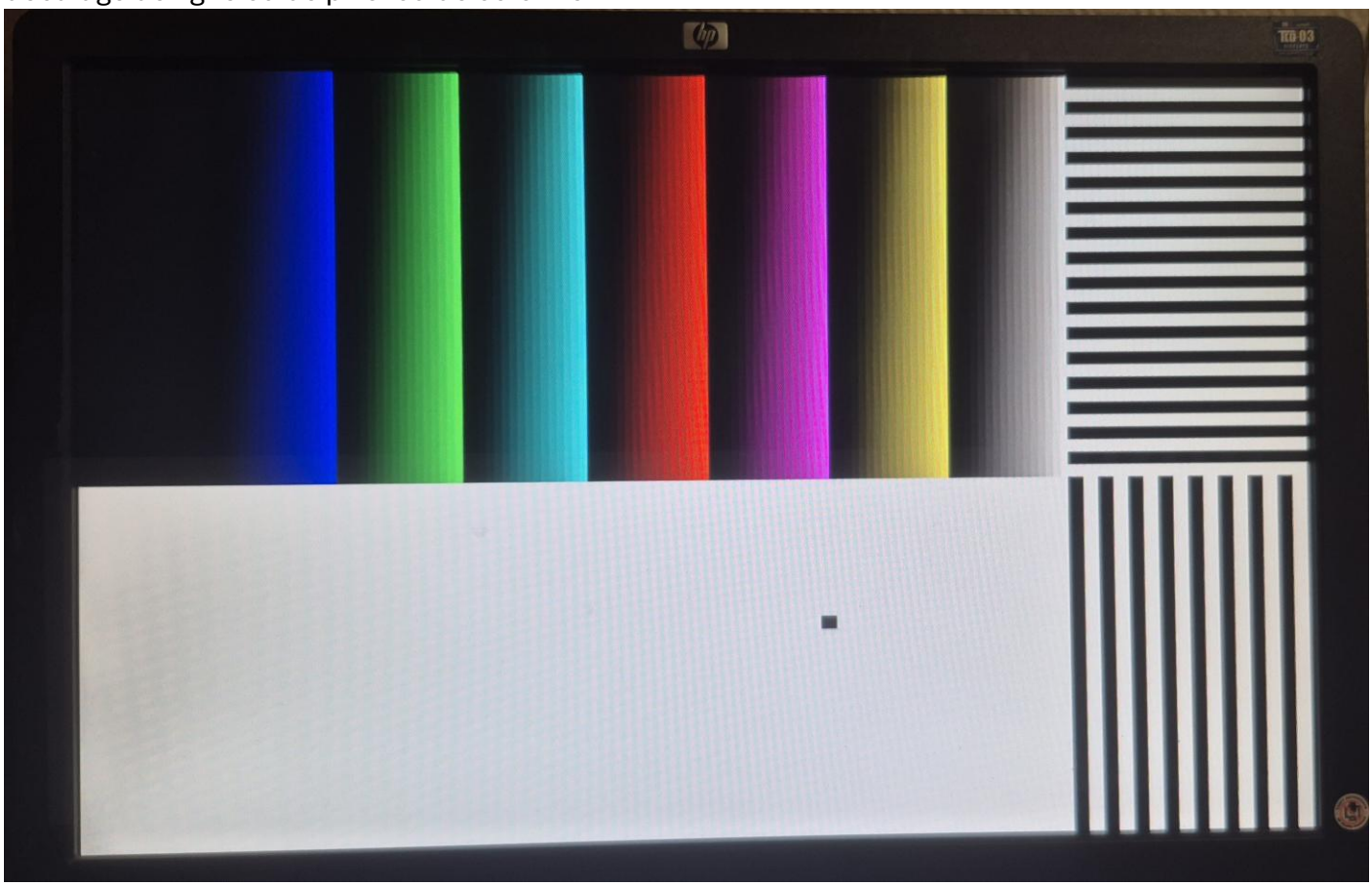


hsync est à 0 pendant 3.83 us ~ 96*periode(25Mhz)
hsync à une période de 32us = 800*periode(clk)

Pattern generator

3.4. Démo

L'image est affichée sur l'ecran , on vérifier que celle-ci et stable et ne présenta visuellement de décalage de ligne ou de pixel ou de colonne



L'image ne présente pas de décalage de pixel visible.
Le point noir dans le cadre en bas à gauche se déplace dans ce cadre.

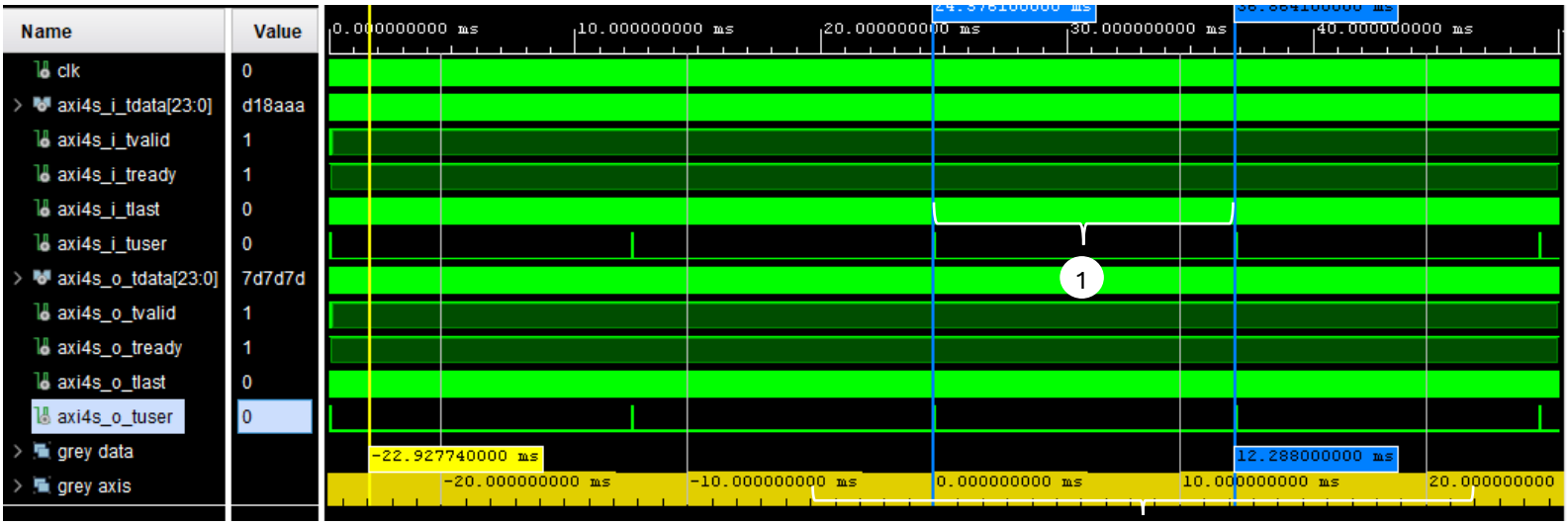
4. Le Filtre gris

4.1. Validation du bloc - simulation

4.1.1. Test bench

Le test bench utilisé consiste à utiliser 3 générateurs de trame axis4s fournit (axi4sdriver. vhd)
axi4sdriver fournit une trame axi4s à partir d'un fichier texte en entrée.
Les 3 fichiers texte doivent contenir un table de 640*480 mots de 8bit en décimal
Afin de faciliter la validation du transfert des données, les fichiers texte utilisés en entrée ont été remplis de mots de 8 bits générés de façon aléatoire et de sorte que 2 nombres consécutifs ne soient pas identiques

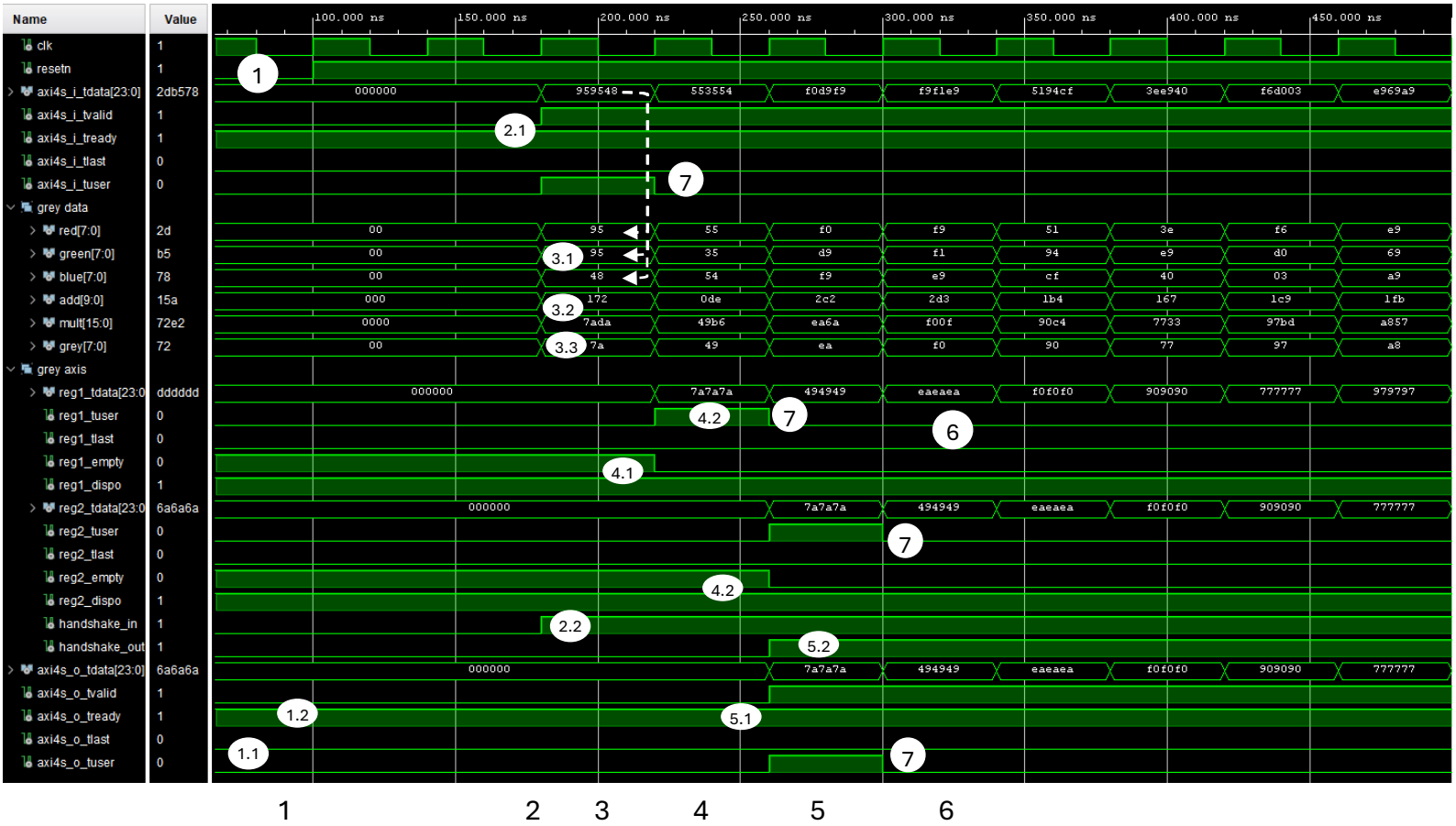
4.1.2. Validation fonctionnelle



1. Le filtre gris transmet les données reçues sur l'axi stream d'entrée sur l'axis stream de sortie après avoir calculé la valeur de gris a transmettre
   Chaque frame est délimitée par le signal tuser .
   Une frame est émise toutes les 12.288ms = 640*480*periode(25Mhz)

Le Filtre gris

4.1.3. Démarrage après reset et premier transfert de donnée

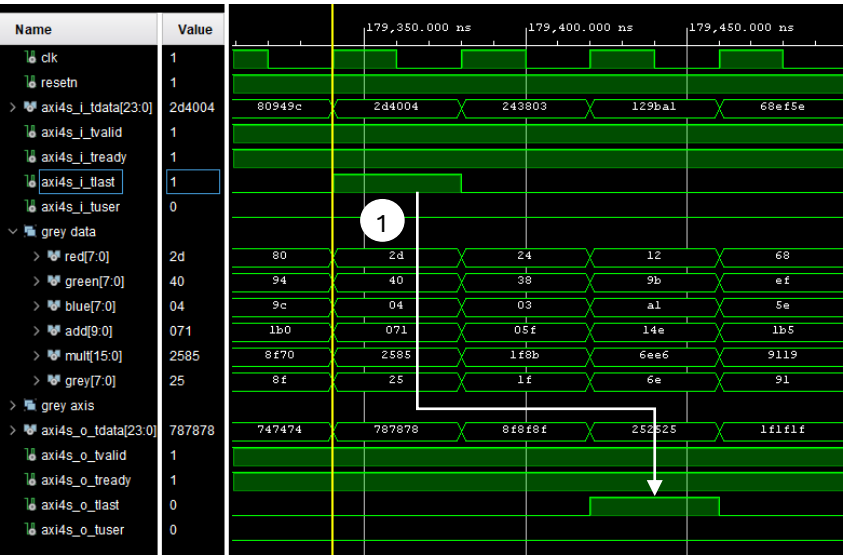


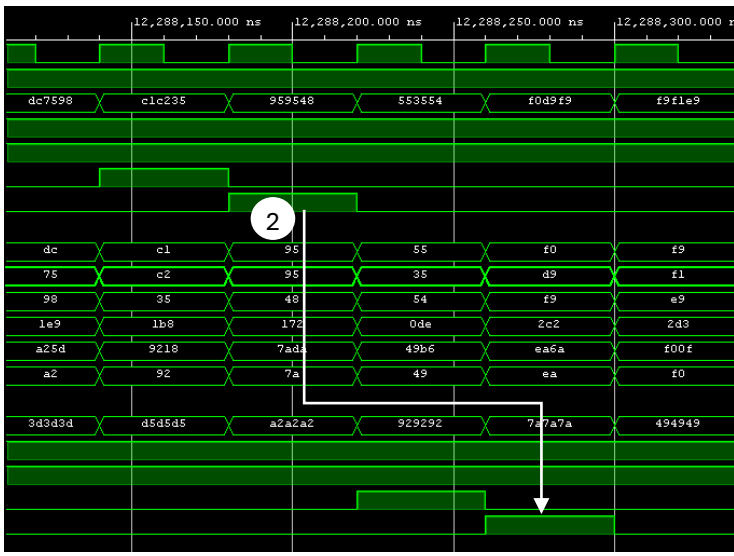
1. Au reset:
   1.1 Les signaux de sortie de contrôle axi4s_o_... sont à zéro: tvalid, tuser, tlast.
   1.2 Le signal tready est à 1 pour indiquer que l'axi slave est prêt pour recevoir de la donnée.
2. La transmission commence après le lever du reset quand le générateur de flux axis_driver commence à envoyer des données
   2.1 axi4s_i_tvalid passe à 1
   Le filtre était prêt à recevoir axi4s_i_tready = 1
   Le handshake_in = 1: le filtre peut recevoir des données à calculer
3. 3.1Les données sont envoyées sur le premier étage de calcul
   3.2 le calcul est effectué sur le même cycle en combinatoire
   3.3 la donnée grise est prête
   tdata est généré en concaténant les trois couleurs générées par le pattern.
4. 4.1 Le premier registre est vide et prêt à recevoir une donnée
   4.2 Les données sont stockées dans le premier registre
   4.3 Les second registre est vide et disponible pour recevoir une donnée
5. 5.1 En sortie l'étage suivant est prêt a recevoir une donnée axi4s_o_tready = 1
   5.2 Une donnée était disponible dans les registre , elle est envoyée: handshake_out = 1
6. Les données suivante sont envoyée sans interruption
7. Les signaux tuser et tlast sont transmis en même temps que les données

La latence du filtre gris est de 2.

Le Filtre gris

4.1.4. Changement de ligne , changement de frame





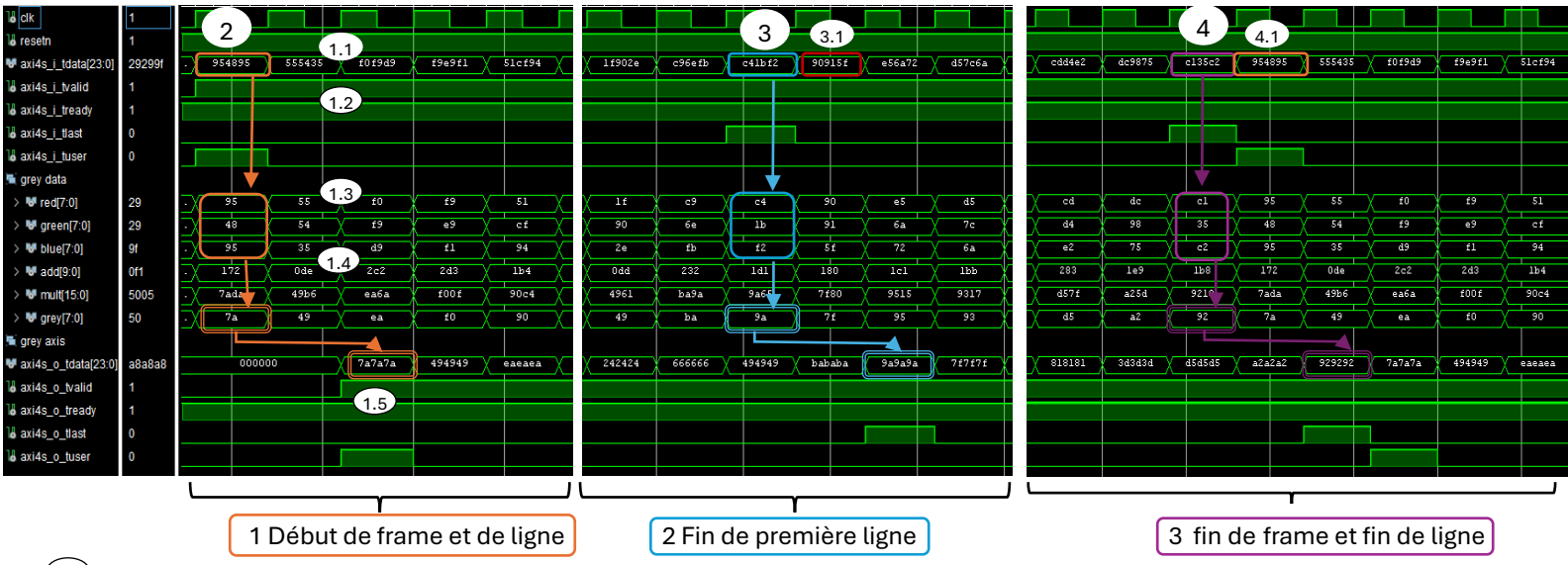
1 changement de ligne

2 changement de frame

1. Au changement de ligne :
   Le signal axi4s_i_tlast passe à 1 sur le dernier pixel de la ligne, il est transmis en même temps que la donnée sur le signal de sortie axi4s_o_tlast
2. Au changement de frame :
   Le signal axi4s_i_tuser passe à 1 sur le dernier pixel de la ligne, il est transmis en même temps que la donnée sur le signal de sortie axi4s_o_tuser

Le Filtre gris

4.1.5. Calcul du filtre gris



1.1

image_alea_r_hexa_corner.txt — entrées red

```
95 55 F0 F9 51 .. 20 6A 1F C9 C4
90 E5 D5 67 80 .. 23 D4 5E 75 5B
D2 45 20 11 31 .. 8  8F DD 5F 91
DB 44 EC E  32 .. 30 D6 32 D1 DC
B7 77 47 A3 A5 .. 81 4C AA 6D BF
.. .. .. .. .. .. .. .. .. .. ..
E  E8 E7 5F 9C .. E3 72 17 3E C
76 3F BD A  C8 .. 5  9B F3 2B E7
B1 2  61 A6 25 .. C0 A8 1F 5A 58
A8 E7 D5 33 4D .. 91 C2 F8 A5 D0
26 8A 7B 48 D0 .. AA 86 CD DC C1
```

image_alea_g_hexa_corner.txt — entrée green

```
48 54 F9 E9 CF .. 41 9A 90 6E 1B
91 6A 7C C9 6E .. 86 3B 70 A9 63
25 62 75 FC 5D .. DC 91 6D E7 55
62 4F D8 E4 24 .. 55 D5 97 C3 24
CF 58 97 2D 74 .. 9E 26 4  5F 6E
.. .. .. .. .. .. .. .. .. .. ..
3C C9 C6 82 3B .. 35 DF 6E AA 30
D4 58 C8 A6 8D .. F9 35 7C D6 DC
BF 8D 76 E5 10 .. D7 B0 36 7F 1E
29 11 22 C0 49 .. 70 BE 6C 91 E7
BD 28 99 FD 8A .. A1 E  D4 98 35
```

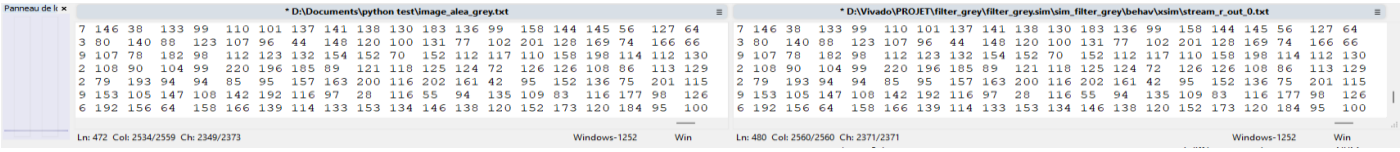
image_alea_b_hexa_corner.txt — entrée blue

```
95 35 D9 F1 94 .. E  30 2E FB F2
5F 72 6A DA 86 .. 18 33 30 AF 3A
24 19 5B 2C 4  .. C1 D4 70 DE D
FD 6E 57 1  44 .. E2 B1 3F A8 C4
4F C8 D  7D C9 .. 33 91 D0 98 41
.. .. .. .. .. .. .. .. .. .. ..
61 45 26 62 60 .. 2F 57 6A 10 A3
C2 24 74 A2 BC .. 41 79 EA 64 3D
C6 E8 B6 9F 4B .. 9B 93 42 CD DB
FB C3 3  74 4D .. C2 A4 CA C4 7B
E1 AC A0 3E B6 .. 3B 25 E2 75 C2
```

image_alea_grey_hexa_corner.txt — sortie grey

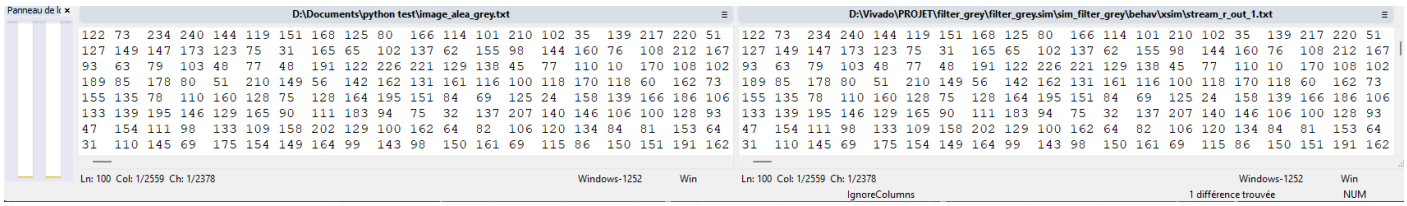
```
7A 49 EA F0 90 .. 24 66 49 BA 9A
7F 95 93 AD 7B .. 40 6A 54 99 52
5D 3F 4F 67 30 .. 8B A6 92 B5 50
BD 55 B2 50 33 .. 77 C8 57 BD 96
9B 87 4E 6E A0 .. 70 55 7E 76 79
.. .. .. .. .. .. .. .. .. .. ..
38 A6 9B 6B 67 .. 6C 8C 4F 52 4A
AD 3E A7 70 AF .. 69 6D C7 76 AA
BB 7C 83 B7 2A .. BA A3 32 8C 6F
98 93 53 77 4B .. 95 B5 B9 A8 BA
96 74 90 80 AF .. 81 3D D5 A2 92
```

1. Déroulement d'une transmission de donné
   1.1 Préparation du signal axi4s_i par le axi4s_driver : lecture des 3 fichiers texte fournis et creation du signal axi4s_i_tdata
   1.2 Au handshake_in , la donnée est utilisée
   1.3 Répartition de axi4s_i_tdata sur les trois signaux red, green, blue du filtre
   1.4 Calcul de la valeur grise
   1.5 Au handshake_out , la donnée est disponible donc utilisée en sortie
2. En début de frame : le premier pixel des fichiers image source est bien lue et envoyé avec son tuser
3. En fin de ligne : le dernier pixel de la première ligne des fichiers image source est bien lue et envoyé avec son tlast
   3.1 La donnée suivante lue et transmise est le premier pixel de la seconde ligne
4. En fin de frame : le dernier pixel de la dernière ligne des fichiers image source est bien lue et envoyé avec son tlast
   4.1La donnée suivante lue et transmise est le premier pixel de la prochaine frame
5. Afin de vérifier que la transmission s'est bien effectuée sur touts les pixel de la frame, on compare le ficher résultat de la simulation avec le fichier résultat théorique calculé.
   L'outil winmerge permet de comparer ces fichiers.
   La comparaison se fait sur les 2 premières frames générées par les test bench



Première frame générée sans erreur. S'il y avait une erreur elle serait affichée dans la colonne de gauche

Le Filtre gris

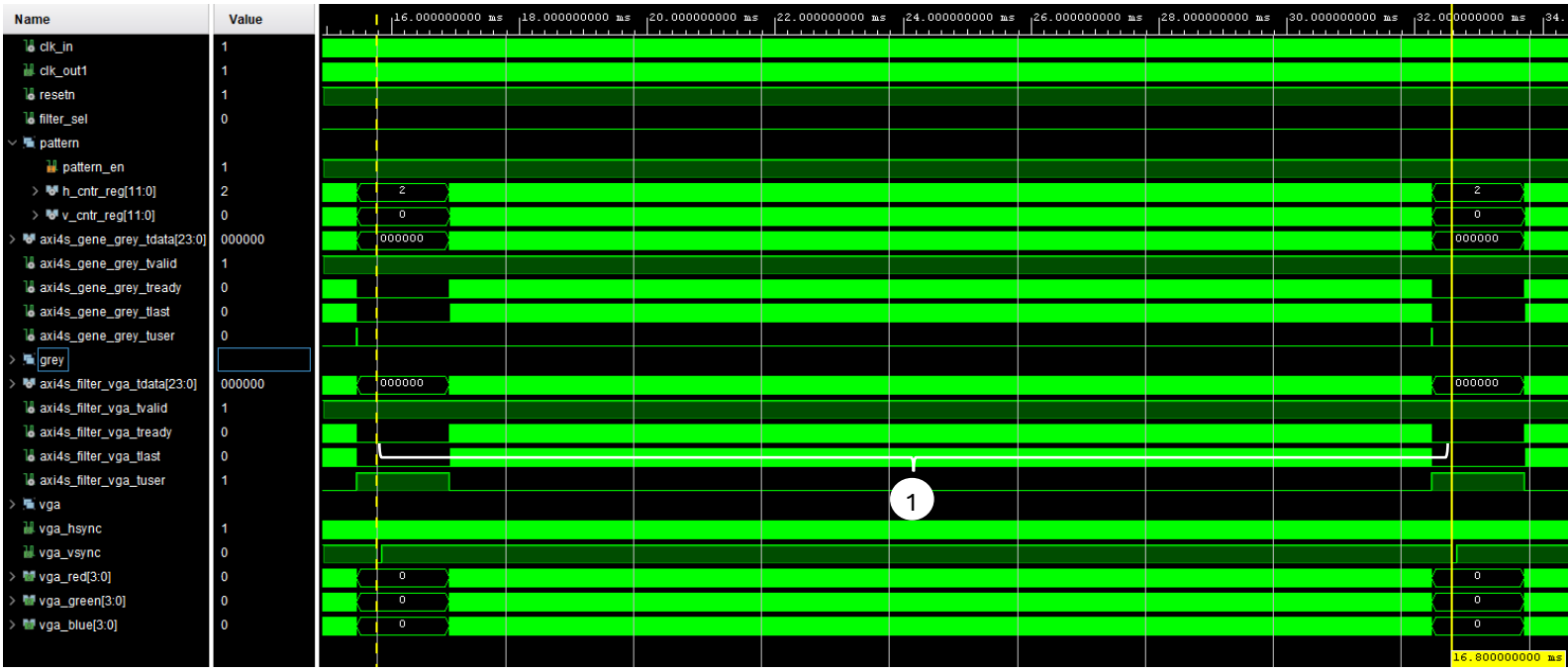


Deuxième frame générée sans erreur.

4.1. Validation du filtre gris – Intégration dans le top

Les blocs présent pour cette simulation sont la pll, le pattern generator, le filtre gris et le convertisseur axi4s_to_vga.

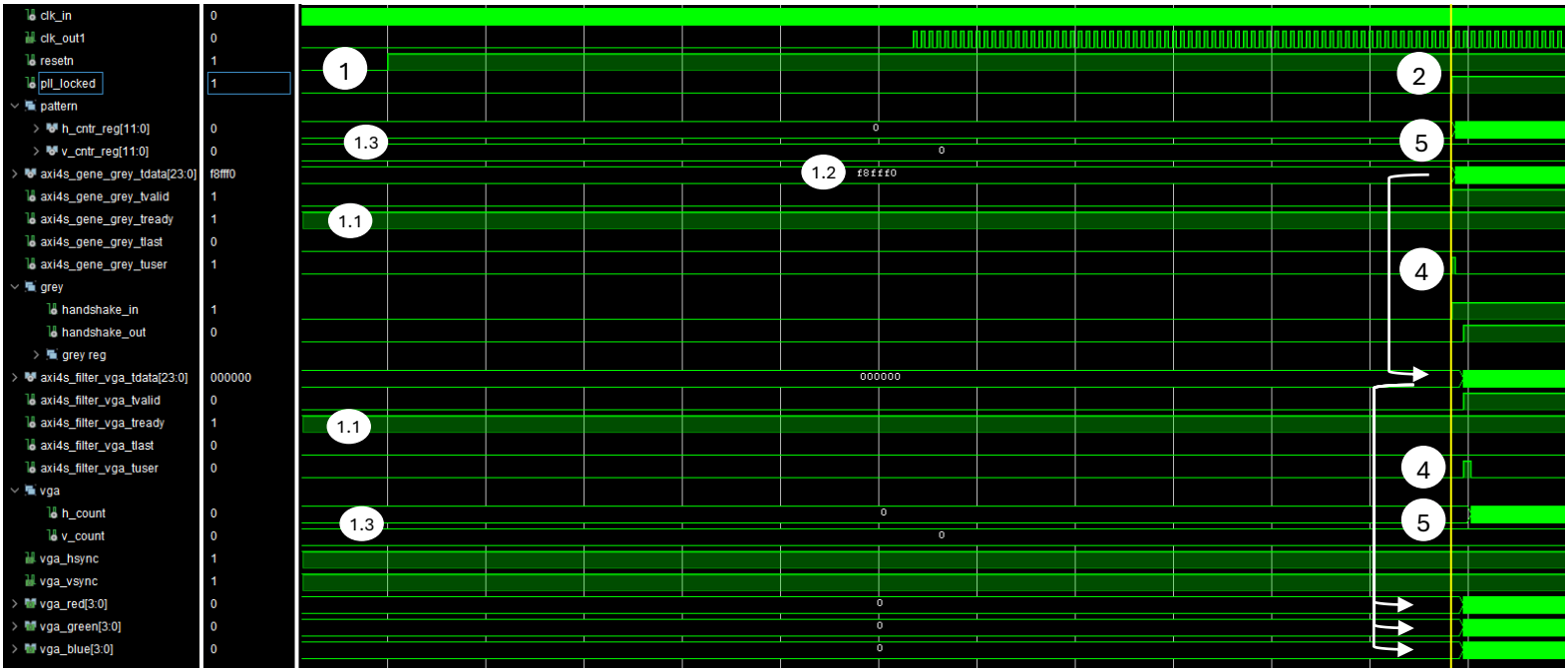
4.1.1. Validation fonctionnelle



1. Le pattern generator crée une frame transférée en axi stream sur un filtre gris
   Le signal filtré en envoyé sur le convertisseur axi4s_to_vga
   Cette communication respecte le protocole VGA
   Une frame est émise toutes les 16.8ms = 800*525*periode(25Mhz)

Le Filtre gris

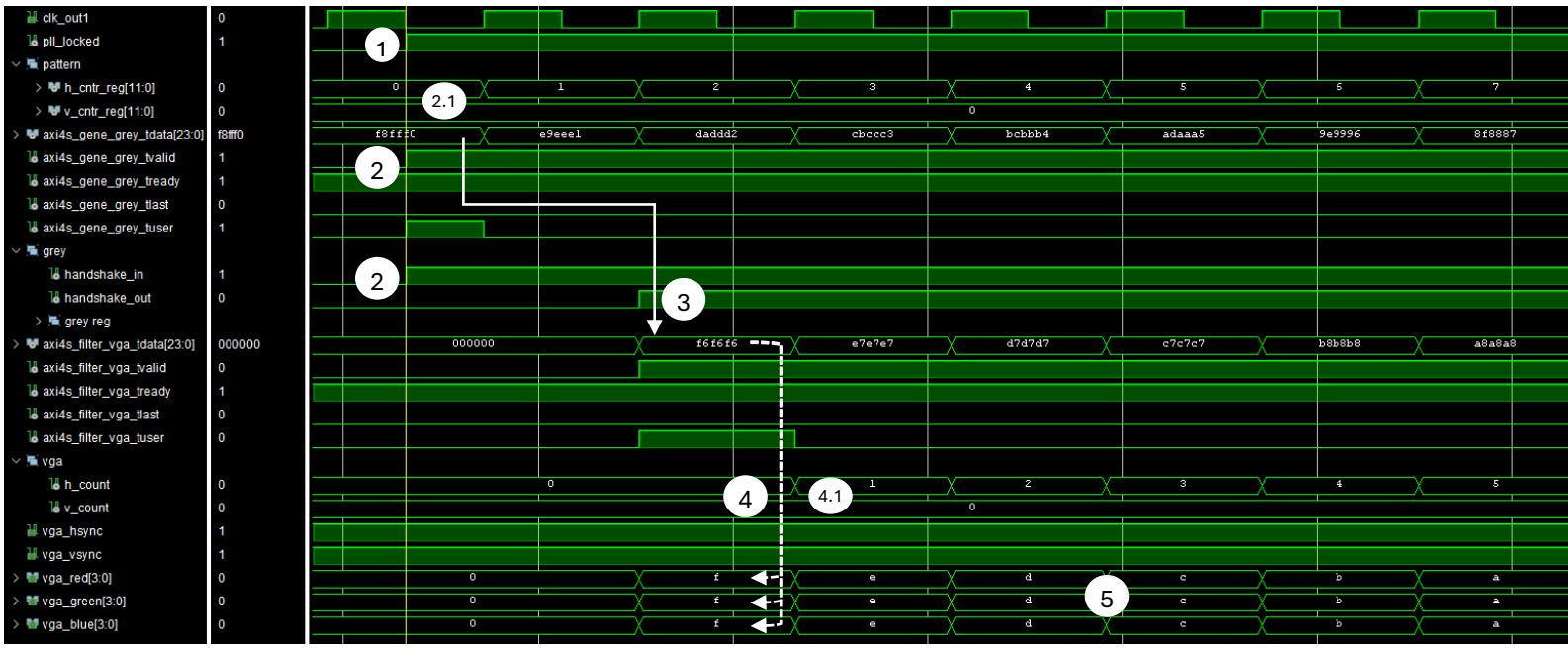
4.1.2. Démarrage après reset



1. Au reset, le signaux de sortie de contrôle axi stream sont à zéro: tvalid, tuser, tlast.
   1.1 Les signaux tready est à 1 pour indiquer que l'axi slave est prêt pour recevoir de la donnée au démarrage.
   1.2 Les données sont générées dans la pattern generator mais ne sont pas envoyées. iMais rien n'est envoyé en sortie du filtre gris
   1.3 Les compteurs sont à 0.
2. Le pattern generator est mis en marche quand la pll a démarré sur pll_locked
   Le tvalid suit pattern_en et indique que les données sont prêtes à être envoyées
3. Les données générées par le pattern sont envoyées au filtres puis au convertisseur axi4s_to_vga
4. tuser passe à 1 sur le premier pixel de la frame
5. les compteurs du pattern generator et du vga démarrent, notamment les compteurs de pixel , les compteurs de lignes s'incrémenterons plus tard

Le Filtre gris

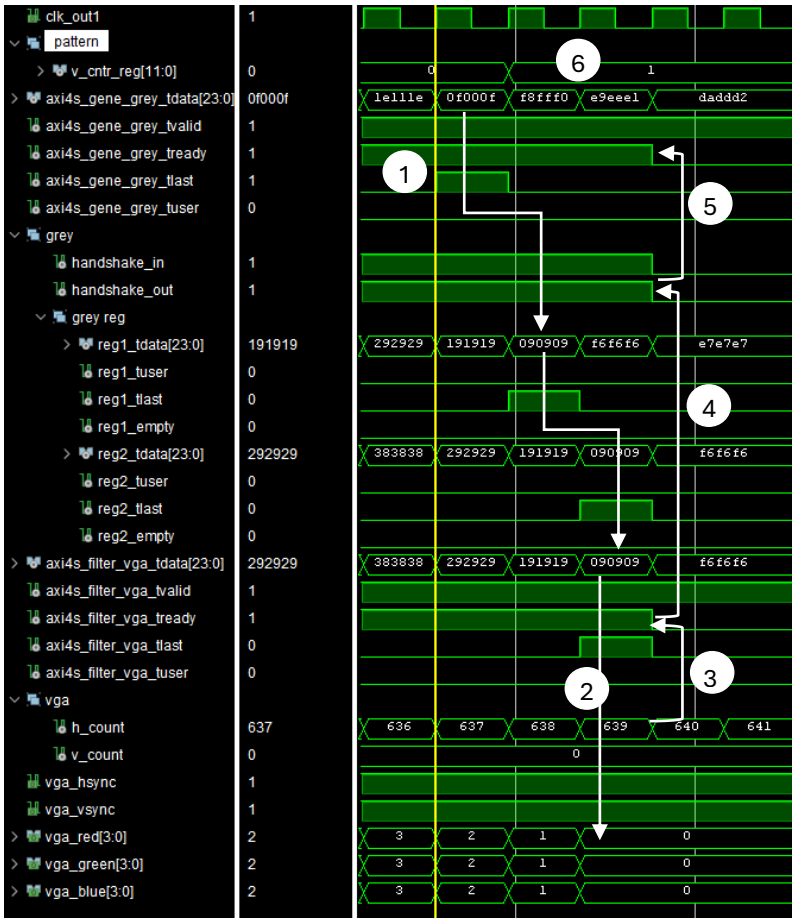
4.1.3. Communication axi stream entre les différents blocs



1. Les données sur la pattern generator sont prêtes dès que la pll est lockée.
2. Le pattern monte son axi4s_gene_grey_tvalid =1
   comme le filtre est prêt à recevoir axi4s_gene_grey_tready= 1
   => Le handshake_in du filtre est OK: La donnée est reçue
   2.1 : les compteurs du pattern generator commencent s'incrémenter
3. Les donnée sont calculées puis transmises 2 coups d'horloge plus tard sur le handshake_out du filtre.
4. La donnée est envoyée au vga sur le même cycle d'horloge
   4.1 : les compteurs du axi4s_to_vga commencent s'incrémenter
5. Les données suivantes sont envoyées au vga sans rupture sur la chaine de données VGA

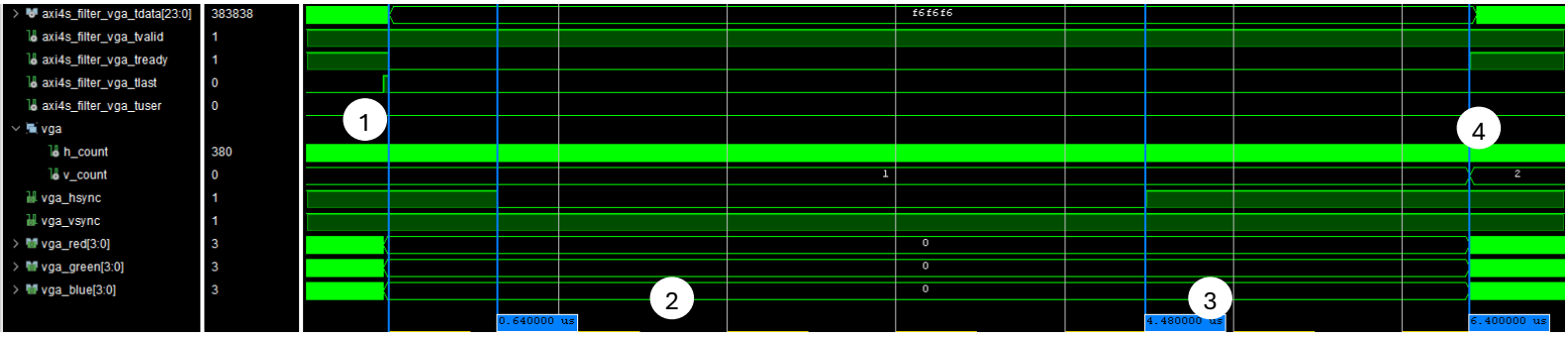
Le Filtre gris

4.1.4. Changement de ligne ,



1. Le dernier pixel de la ligne est généré et envoyé au filtre
2. Le filtre envoie le dernier pixel au vga 2 cycle d'horloge plus tard
3. Le convertisseur axi4_to _vga détecte une fin de ligne et relâche axi4s_filter_vga_tready puis continue son comptage pour la fin de cycle h_sync
4. Le filtre ne peux plus envoyer de données son handshake_out est relâché.
5. Dès que le filtre a ses 2 registres pleins ,il bloque l'acquisition de nouvelle données . axi4s_gene_grey_tready = 0
6. Les 2 premiers pixels de la ligne suivantes ont été calculés et stockés dans les registres du filtre gris

4.1.5. Validation des timings hsync

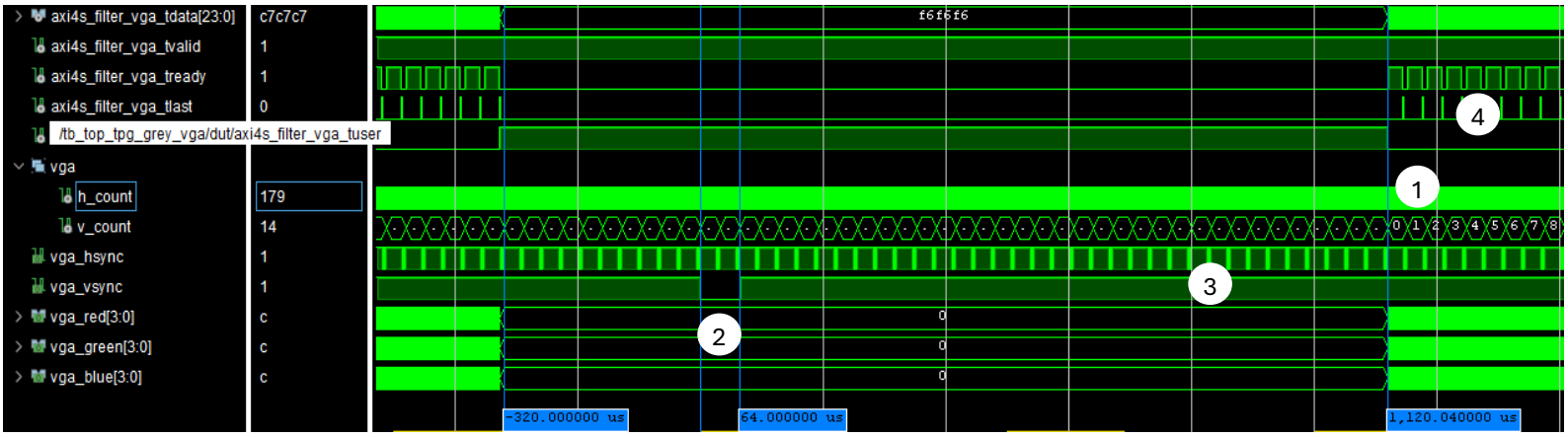


1. Fin de ligne sur le filtre gris : axi4s_filter_vga_tlast = 1
2. Le signal hsync passe à 0 0.64us plus tard soit 16* periode(25Mhz)
3. Le signal hsync reste à 0 pendant 96 periode , il repasse à 1 à 4.48us après la fin de ligne 4.48us = (16+96)* periode(25Mhz)
4. Le protocole VGA dis d'attendre 48 periode supplémentaires avant de lancer la transmission d'un autre ligne

Le signal tready repasse à 1 à t=6.4us après la fin de ligne pour relancer la transmission d'un autre mot. 6.4us = (16+96+48)* periode(25Mhz)

Le Filtre gris

4.1.6. Validation du timings vsync sur un transmission de frame



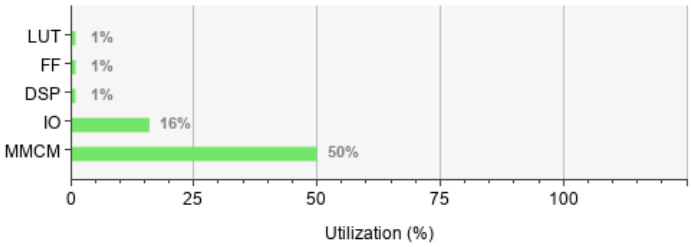
1. Au changement de frame v_count et h_count sont remis à zéro
   Les données commencent à être transférées en commençant par la première ligne (v_count = 0)
2. Le signal vsync attend 10 lignes avant de passer à 0 soit 320us = 800*10*periode(25Mhz)après la fin de frame
3. Le signal vsync reste à 0 pendant 2 lignes, soit 64us = 800*2*periode(25Mhz)
4. La transmission est relancée quand 33 cycles de lignes sont passés à t= 1.120us (800*(2+33))*periode(25Mhz)

4.1. Test sur carte

4.1.1. Synthèse et implémentation : rapport d'utilisation

Name	Slice LUTs (17600)	Slice Registers (35200)	DSPs (80)	Bonded IOB (100)	BUFGCTRL (32)	MMCME2_ADV (2)
top_tpg_grey_vga	195	105	1	16	2	1
u_axi4s_to_vga (axi4s_to_vga)	34	21	0	0	0	0
u_clk_wiz_0 (clk_wiz_0)	0	0	0	0	2	1
u_filter_grey (filter_grey)	17	9	1	0	0	0
u_pattern_generator (pattern_generator)	143	75	0	0	0	0

Resource	Utilization	Available	Utilization %
LUT	188	17600	1.07
FF	105	35200	0.30
DSP	1	80	1.25
IO	16	100	16.00
MMCM	1	2	50.00



Ref Name	Used	Functional Category
FDCE	97	Flop & Latch
LUT2	84	LUT
LUT4	65	LUT
LUT6	46	LUT
CARRY4	37	CarryLogic
LUT1	28	LUT
LUT5	21	LUT
LUT3	20	LUT
OBUF	14	IO
FDPE	8	Flop & Latch
IBUF	2	IO
BUFG	2	Clock
MMCME2_ADV	1	Clock
DSP48E1	1	Block Arithmetic

Toujours le même nombre de registre utilisés pour axi4s_to_vga et pattern generator.
Pour l'implémentation du filtre gris :

9 registres pour les signaux axis
17 LUT pour la logique de calcul principalement
1 DSP surement pour le calcul de la multiplication

4.1.1. Synthèse et implémentation : rapport de timings

Le Filtre gris

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	26,228 ns	Worst Hold Slack (WHS):	0,036 ns	Worst Pulse Width Slack (WPWS):	2,000 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	189	Total Number of Endpoints:	189	Total Number of Endpoints:	112

All user specified timing constraints are met.

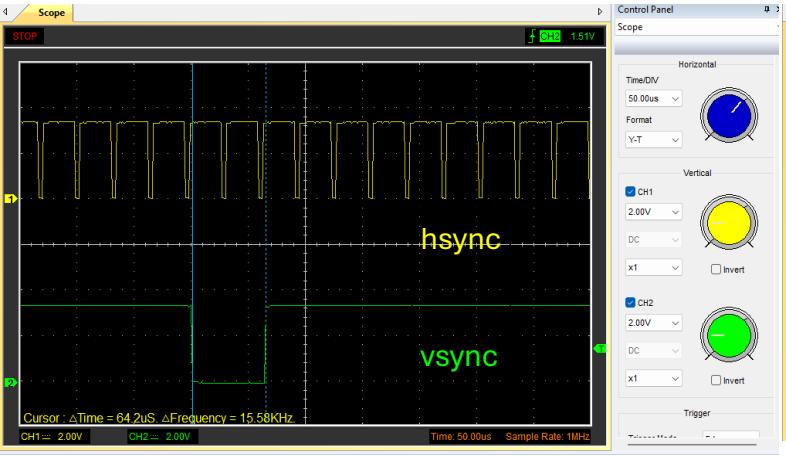
Les contraintes de timing sont respectées avec une horloge à 25Mhz.

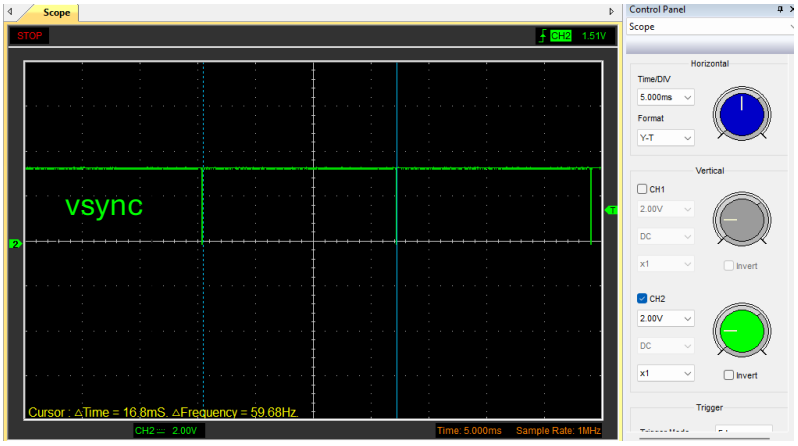
Summary

Name	Path 1
Slack	26.228ns
Source	u_pattern_generator/box_x_reg_reg[3]/C (rising edge-triggered cell FDCE clocked by clk_out1_clk_wiz_0 {rise@0.000ns fall@20.000ns period=40.000ns})
Destination	u_filter_grey/multOp/A[9] (rising edge-triggered cell DSP48E1 clocked by clk_out1_clk_wiz_0 {rise@0.000ns fall@20.000ns period=40.000ns})
Path Group	clk_out1_clk_wiz_0
Path Type	Setup (Max at Slow Process Corner)
Requirement	40.000ns (clk_out1_clk_wiz_0 rise@40.000ns - clk_out1_clk_wiz_0 rise@0.000ns)
Data Path Delay	9.645ns (logic 3.879ns (40.218%) route 5.766ns (59.782%))
Logic Levels	12 (CARRY4=7 LUT2=1 LUT3=1 LUT4=2 LUT6=1)
Clock Path Skew	-0.039ns
Clock Un...rtainty	0.160ns

Le chemin critique est un chemin entre les données du pattern generator et la logique de calcul du filtre gris

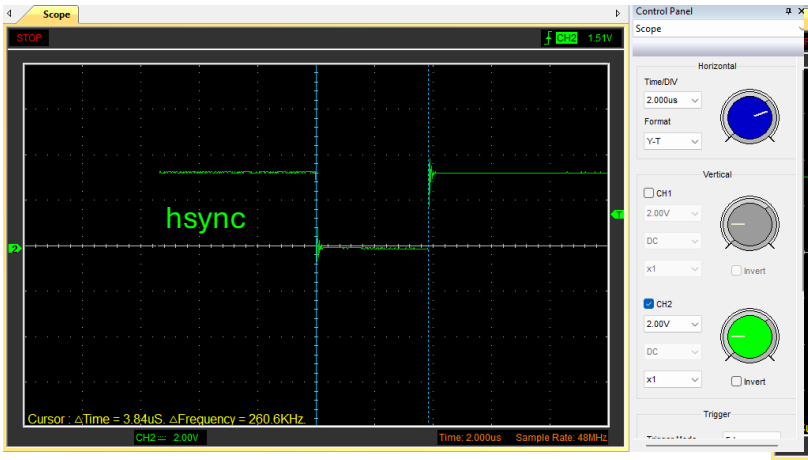
4.1.1. Mesure sur oscilloscope

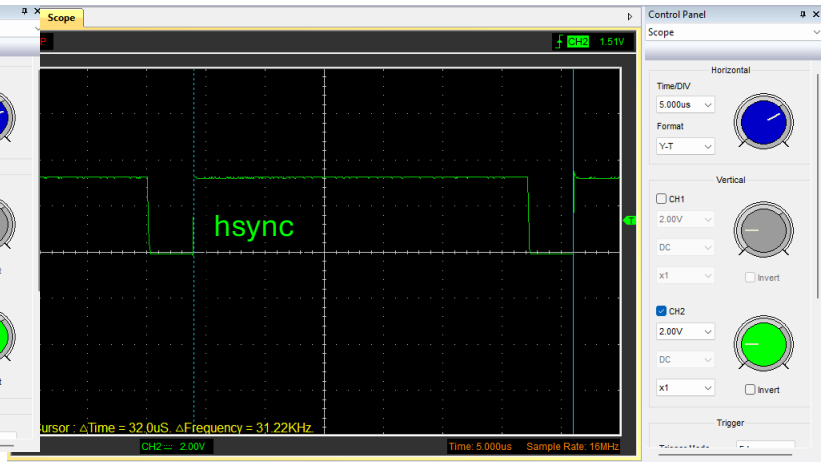




Le signal vsync passe à 0 toutes les 16.8ms = 800*525* periode(25Mhz)
le signal vsync reste à 0 pendant 2 cycles hsync soit 64us= 800*2*periode(25Mhz)

Le Filtre gris

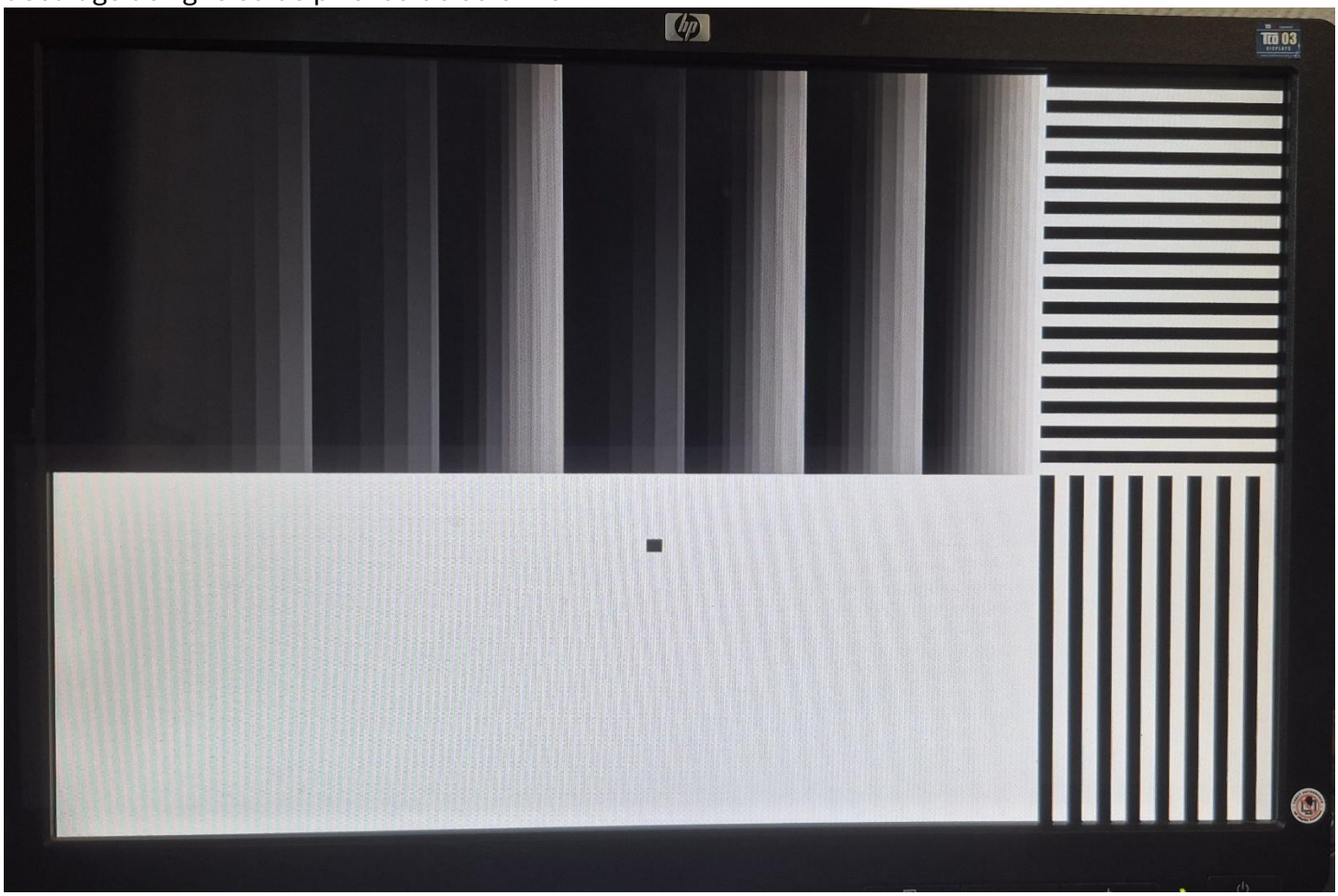




hsync est à 0 pendant 3.83 us ~ 96*periode(25Mhz)
hsync à une période de 32us = 800*periode(clk)

4.1. Démo

L'image est affichée sur l'écran , on vérifier que celle-ci et stable et ne présenta visuellement de décalage de ligne ou de pixel ou de colonne



Le pattern couleur du départ a bien été converti en niveau de gris
L'image ne présente pas de décalage de pixel visible.
Le point noir dans le cadre en bas à gauche se déplace dans ce cadre.

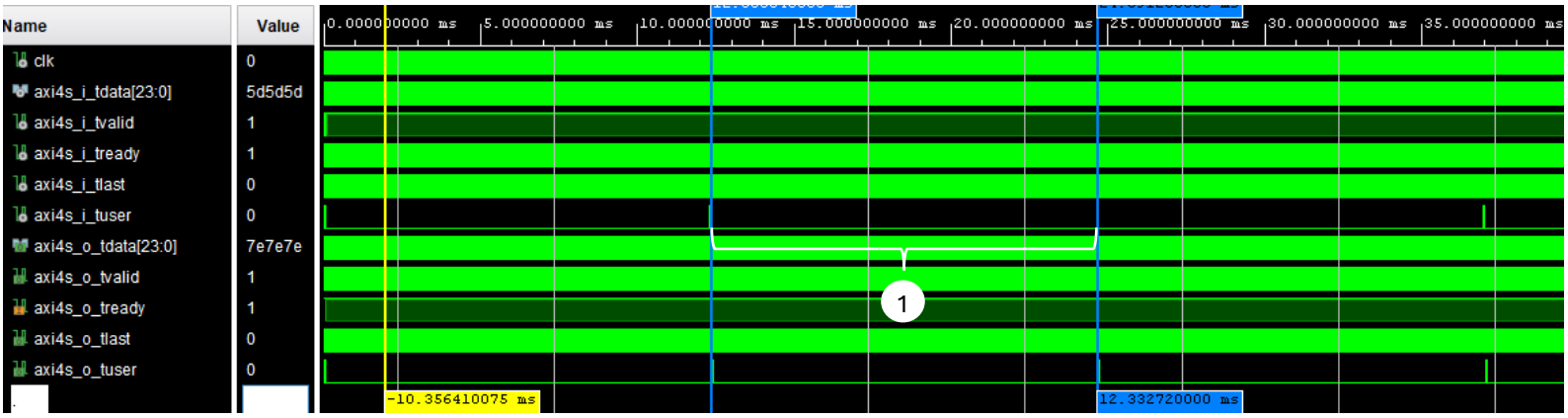
5. Le Filtre de gauss

5.1. Validation du bloc - simulation

5.1.1. Test bench

Le test bench utilisé consiste à utiliser 3 générateurs de trame axis4s fournit (axi4sdriver. vhd)
axi4sdriver fournit une trame axi4s à partir d'un fichier texte en entrée.
Les 3 fichiers texte doivent contenir un table de 640*480 mots de 8bit en décimal
Les 3 fichiers d'entrée sont identiques et comportent le résultat du filtrage gris.
Afin de faciliter la validation du transfert des données, les fichiers texte utilisés en entrée ont été remplis de mots de 8 bits générés de façon aléatoire et de sorte que 2 nombres consécutifs ne soient pas identiques

5.1.2. Validation fonctionnelle

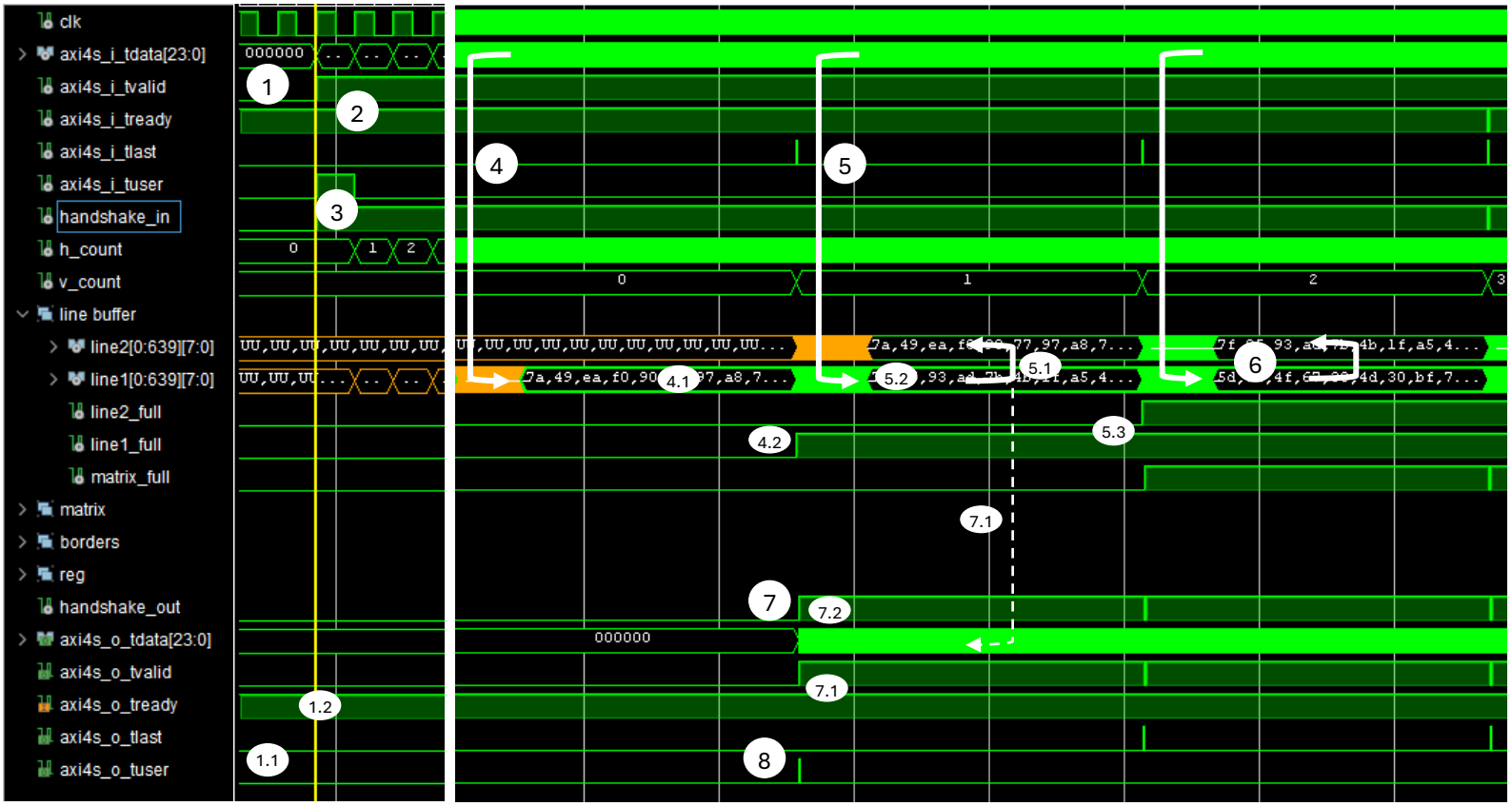


1. Le filtre gauss transmet les données reçues sur l'axi stream d'entrée sur l'axis stream de sortie après avoir calculé la valeur de gauss à transmettre
   Chaque frame est délimitée par le signal tuser .
   Une frame est émise toutes les 12.33272ms = (640*480) + 1118*periode(25Mhz)

Les 1118 périodes supplementaires correspondent au delais introduits par l'architecture du filtre de gauss.: 640 cycle pour le delai d'attente de remplissage du line buffer 1
478 cycle pour le transfert du dernier pisel de chaque lignes sauf pour la première et dernière ligne de la frame.

Le Filtre de gauss

5.1.3. Démarrage après reset et premier transfert de données

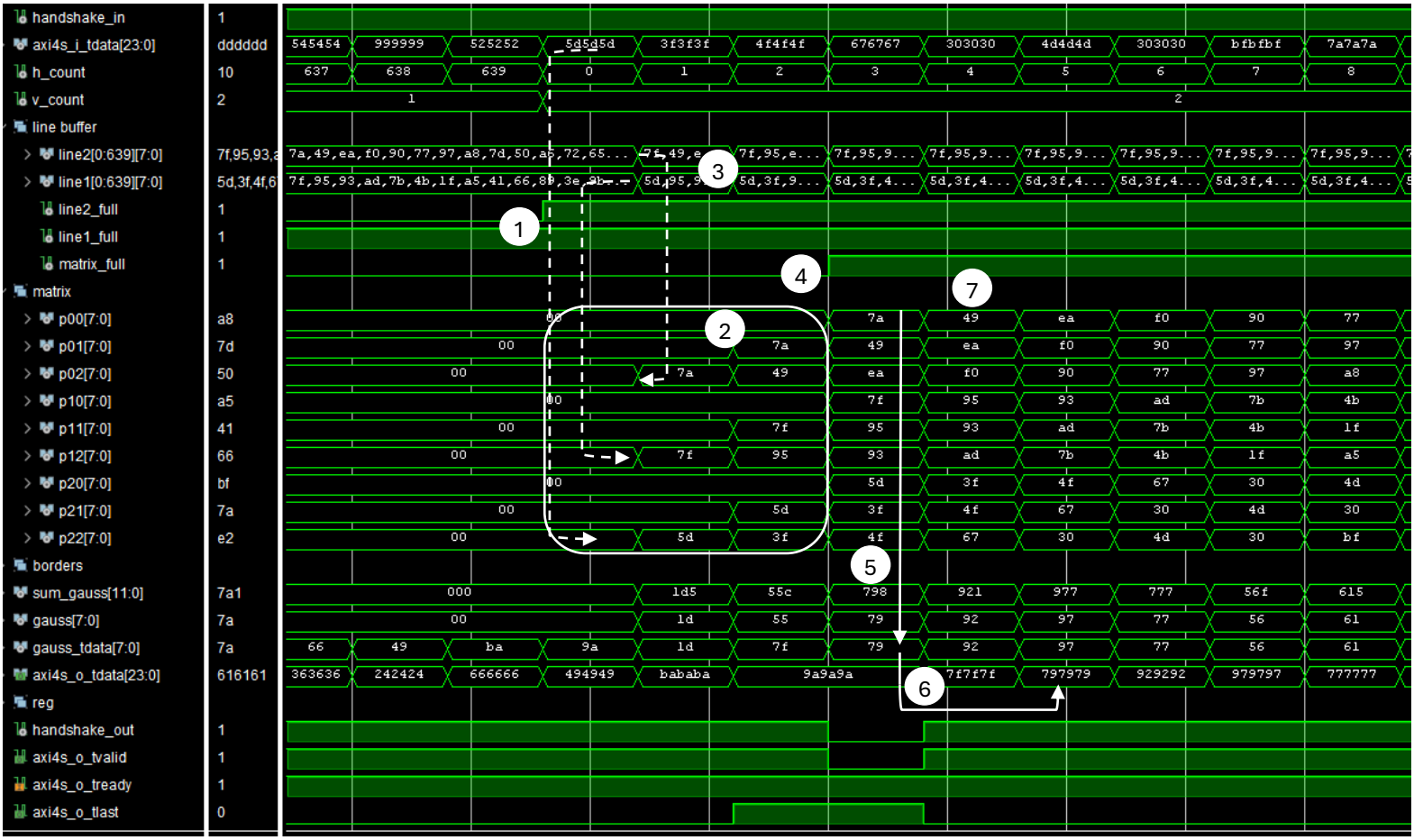


zoom reset

1. Au reset:
   1.1 Les signaux de sortie de contrôle axi4s_o_... sont à zéro: tvalid, tuser, tlast.
   1.2 Le signal tready est à 1 pour indiquer que l'axi slave est prêt pour recevoir de la donnée.
2. La transmission commence après le lever du reset quand le générateur de flux axis_driver commence à envoyer des données
   - axi4s_i_tvalid passe à 1
   - Le filtre était prêt à recevoir axi4s_i_tready = 1
   - Le handshake_in = 1: le filtre peut recevoir des données
3. La première donnée est identifiée par axi4s_i_tuser = 1
4. La première ligne est lue et stockée dans le line1 buffer
   4.1 le line1 buffer se rempli
   4.2 jusqu'à ce qu'il soit plein à la fin de la première ligne lue
5. La seconde ligne est lue et stockée dans le line1 buffer
   5.1 Le line1 buffer 1 est transféré dans le line2buffer
   5.2 Le line1buffer se rempli avec les données lues sur axi4s_i_data
   5.3 A la fin de lecture de la seconde ligne re données les deux line buffers sont pleins
6. Ce remplissage des buffers continue sur les autres lignes lues
7. Dès que le premier line1 buffer est plein, on peut commencer à transférer les données en sortie du filtre
   7.1 axi4s_o_tvalid passe à 1
   le bloc en sortie était prêt à recevoir axi4s_o_tready = 1
   7.2 Le handshake_out = 1: le filtre peut transférer les données
   La première lignes est envoyée en sortie sans opérations de calcul
8. Le signal axi4s_o_tuser passe à 1 sur le premier pixel de la frame transmis

Le Filtre de gauss

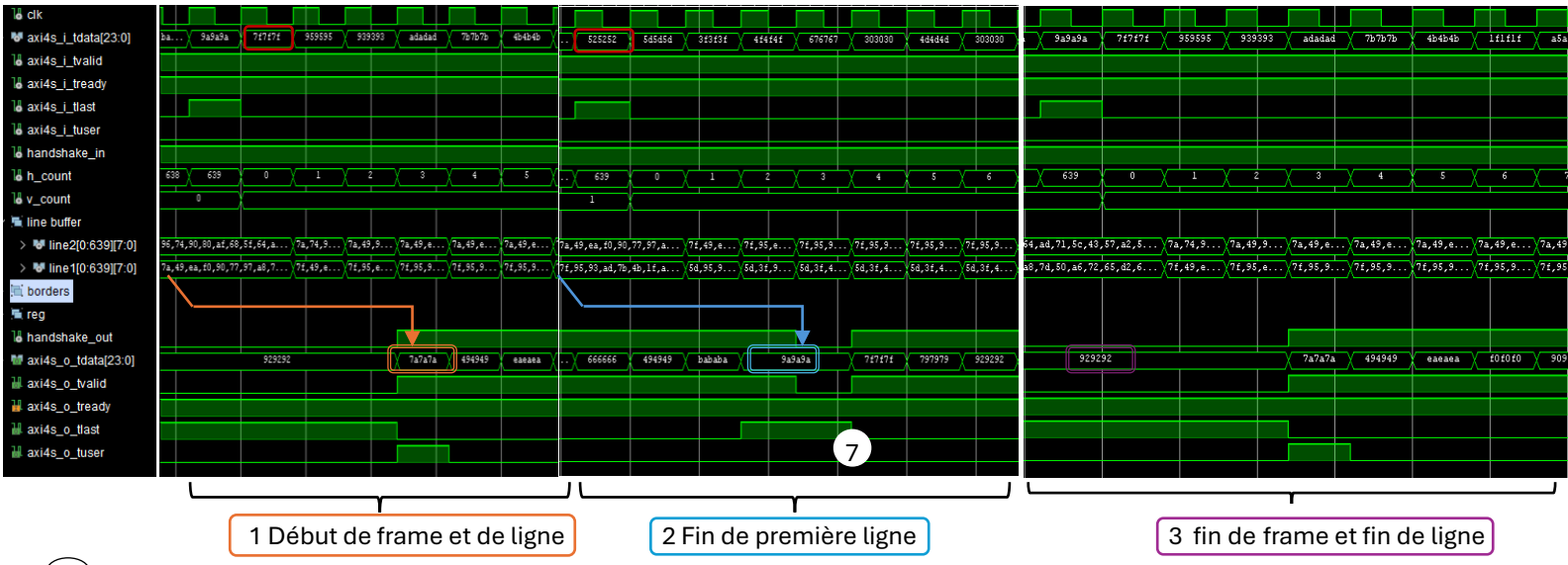
5.1.1. Calcul du filtrage : line buffer et matrice



1. Dès que les 2 lines buffers sont plein , la matrice de calcul peut commencer à se remplir
2. La matrice se remplie avec les contenue de line1, line2 et i_data
3. En même temps les lines buffers continuent de se remplir aussi
4. Après 3 coup d'horloge , La matrice est pleine est pleine
5. Les calculs sont exécutés et pris en compte dans le transfert des données quand la matrice est pleine.
6. La valeur calculée est transmise sur le bus de data 2 coup d'horloge plus tard suivant le protocole des registre du filtre gauss
7. .Le même calcul est éxécuté sur les pixels suivants

Le Filtre de gauss

5.1.2. Calcul du filtre gauss



1.1

image_alea_grey_hexa_corner.txt — entrées grise

7A	49	EA	F0	90	..	24	66	49	BA	9A
7F	95	93	AD	7B	..	40	6A	54	99	52
5D	3F	4F	67	30	..	8B	A6	92	B5	50
BD	55	B2	50	33	..	77	C8	57	BD	96
9B	87	4E	6E	A0	..	70	55	7E	76	79
..	..	..	..	..	..	..	..	..	..	..
38	A6	9B	6B	67	..	6C	8C	4F	52	4A
AD	3E	A7	70	AF	..	69	6D	C7	76	AA
BB	7C	83	B7	2A	..	BA	A3	32	8C	6F
98	93	53	77	4B	..	95	B5	B9	A8	BA
96	74	90	80	AF	..	81	3D	D5	A2	92

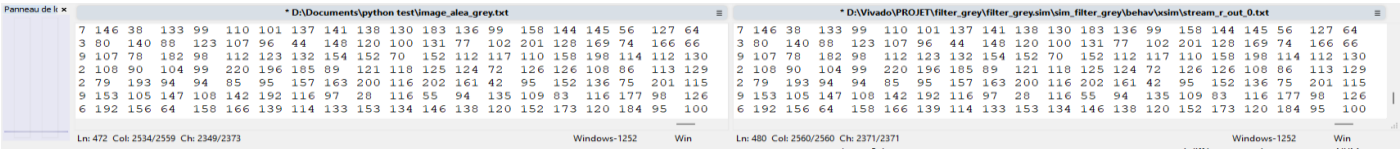
image_alea_gauss_hexa_corner.txt — sortie gauss

7A	49	EA	F0	90	..	24	66	49	BA	9A
7F	79	92	97	77	..	5E	67	78	85	52
5D	6A	6F	68	59	..	7E	89	8D	8D	50
BD	75	6E	62	65	..	80	8B	8A	8F	96
9B	86	7C	78	81	..	6B	72	7C	81	79
..	..	..	..	..	..	..	..	..	..	..
38	79	83	7A	79	..	73	77	75	74	4A
AD	7F	87	85	80	..	7F	81	7F	7B	AA
BB	84	81	7F	6D	..	99	92	86	89	6F
98	86	7A	77	6E	..	99	97	9B	9E	BA
96	74	90	80	AF	..	81	3D	D5	A2	92

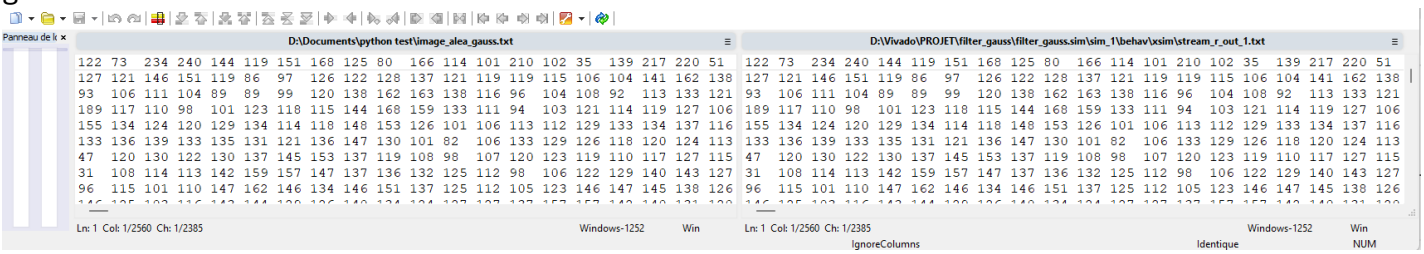
La transimission des données est décalé d'une ligne complète . Il est difficile de lereprésenter sur un chronogramme dans word .
Les checks de debut et fin de ligne ainsi quel es check sur les premières et dernières lignes sont des cas particuliers car sur ces pixels aucun calcul est fait. Sur les pixels de bordure la valeur du pixel initial est recopiée.
Afin de vérifier que la transmission s'est bien effectuée sur touts les pixel de la frame, on compare le ficher résultat de la simulation avec le fichier résultat théorique calculé.

L'outil winmerge permet de comparer ces fichiers.

La comparaison se fait sur les 2 premières frames générées par les test bench



Première frame générée sans erreur. S'il y avait une erreur elle serait affichée dans la colonne de gauche

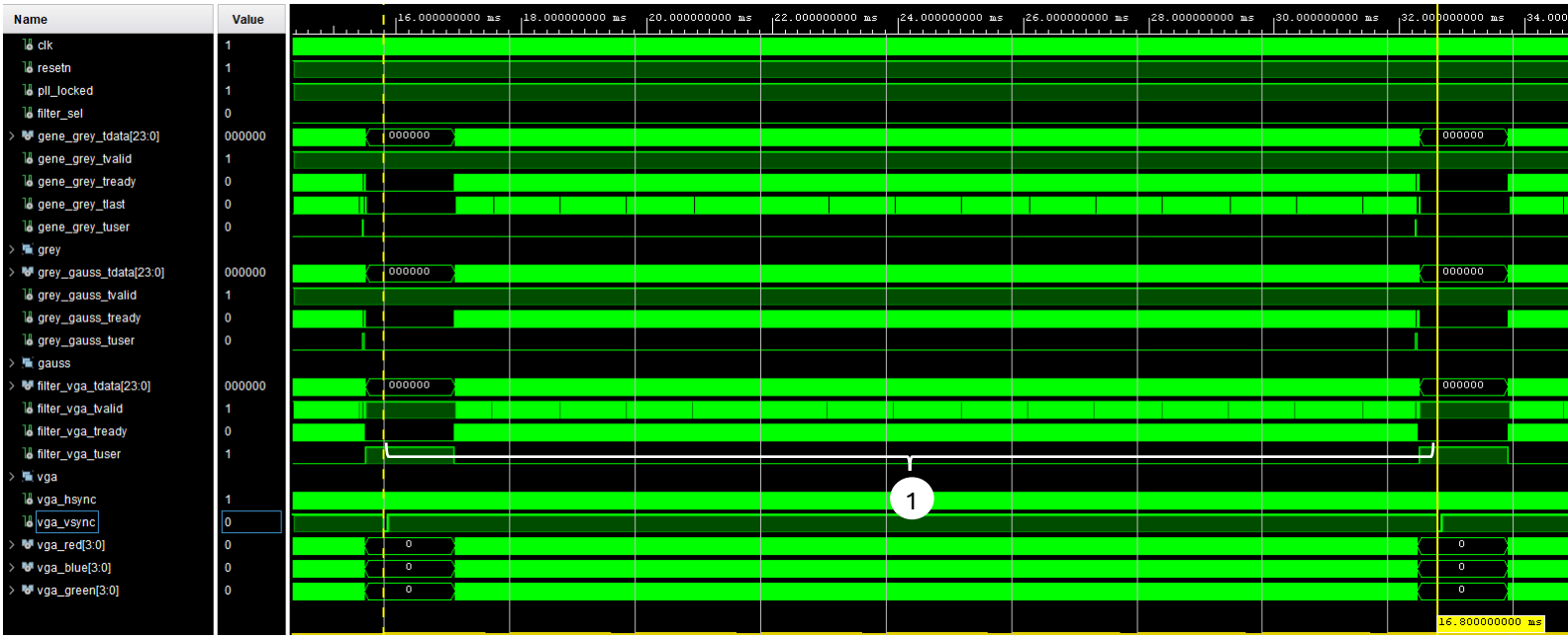


Deuxième frame générée sans erreur.

5.1. Validation du filtre de gauss – Intégration dans le top

Les blocs présent pour cette simulation sont la pll, le pattern generator, le filtre gris, le filtre de gauss et le convertisseur axi4s_to_vga.

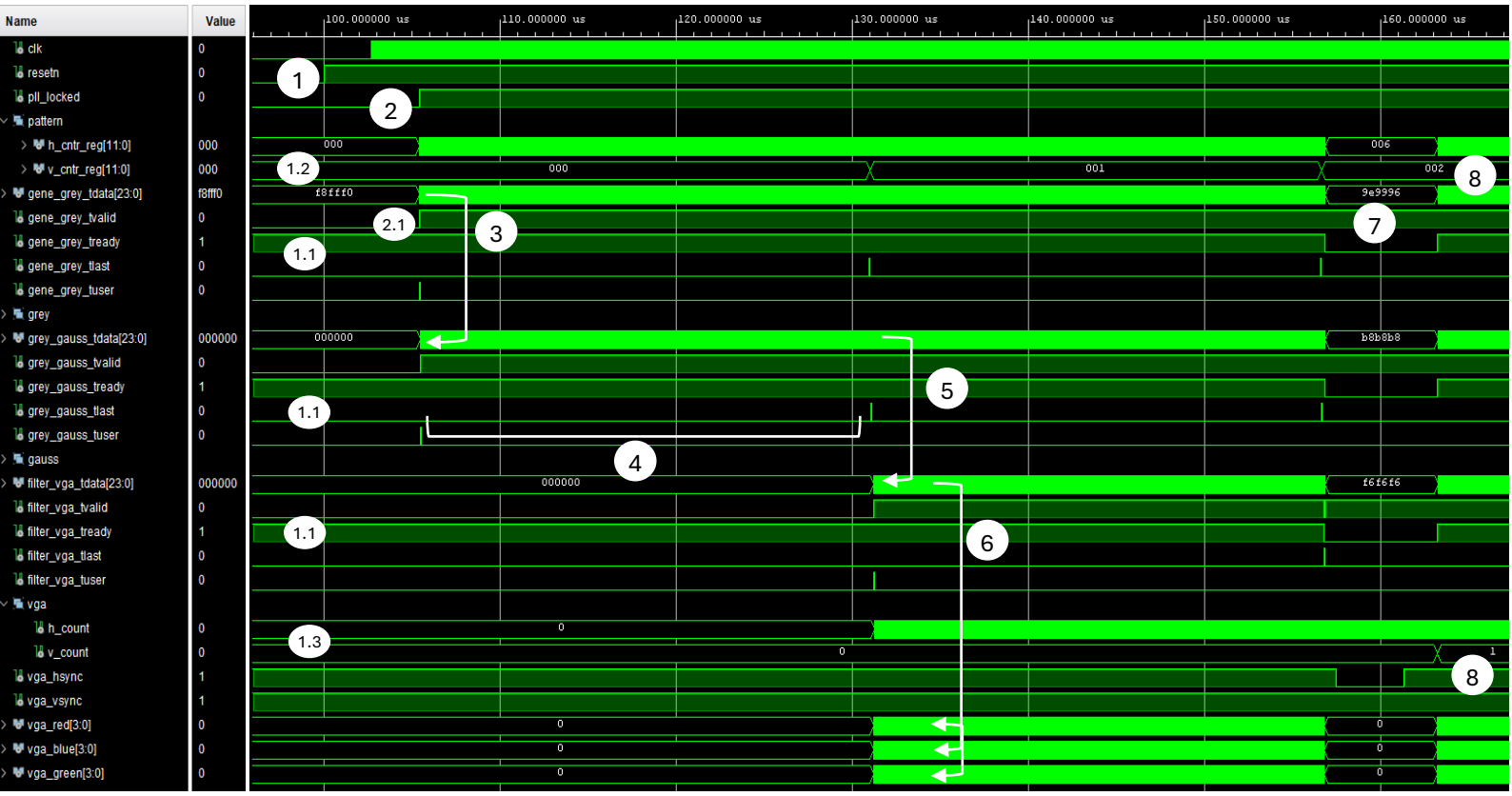
5.1.1. Validation fonctionnelle



1. Le pattern generator crée une frame transférée en axi stream sur le filtre gris, puis sur le filtre gauss
   Le signal filtré en envoyé sur le convertisseur axi4s_to_vga
   Cette communication respecte le protocole VGA
   Une frame est émise toutes les 16.8ms = 800*525*periode(25Mhz)

Le Filtre de gauss

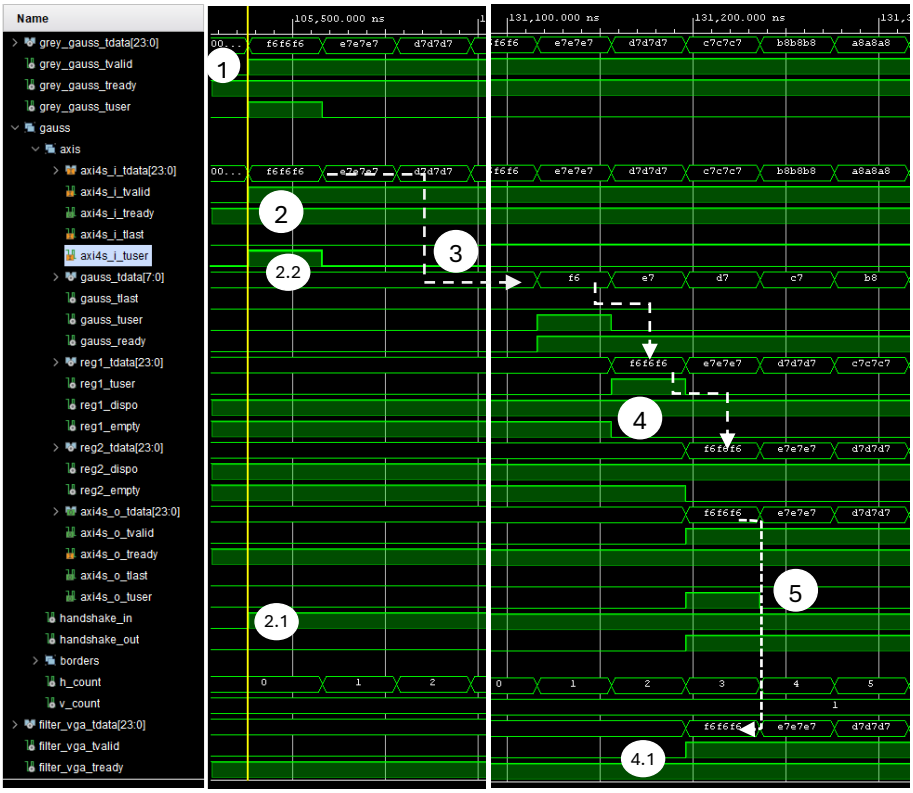
5.1.2. Démarrage après reset



1. Au reset, le signaux de sortie de contrôle axi stream sont à zéro: tvalid, tuser, tlast.
   1.1 Les signaux tready est à 1 pour indiquer que les axi slave seront prêt pour recevoir de la donnée au démarrage.
   1.2 Les compteurs du pattern et du vga sont à 0.
2. Le pattern generator est mis en marche quand la pll a démarré sur pll_locked
   2.1Le tvalid suit pattern_en et indique que les données sont prêtes à être envoyées
3. Les données générées par le pattern sont envoyées au filtre gris sur le handshake gene_grey gene_grey_tready = 1 et gene_grey_tvalid = 1
4. Le filtre de gauss commence à fournir des données au vga un cycle de ligne plus tard
5. Les données générées par le filtre gris sont envoyées au filtre gauss sur le handshake grey_gauss grey_gauss_tready = 1 et grey_gauss_tvalid= 1
6. Les données générées par le filtre gauss sont envoyées au vga sur le handshake filter_vga filter_vga_tready = 1 et filter_vga_tvalid= 1
7. La première ligne est transmise, le pattern generator se met en pause en attendant la fin du protocole vga gene_grey_tready = grey_gauss_tready = filter_vga_tready 0
8. Le protocole vga est terminé, on passe à l'affichage de la deuxième ligne, la pattern generator et toute la ligne de filtrage redémarre pour le calcul de la deuxième ligne

Le Filtre de gauss

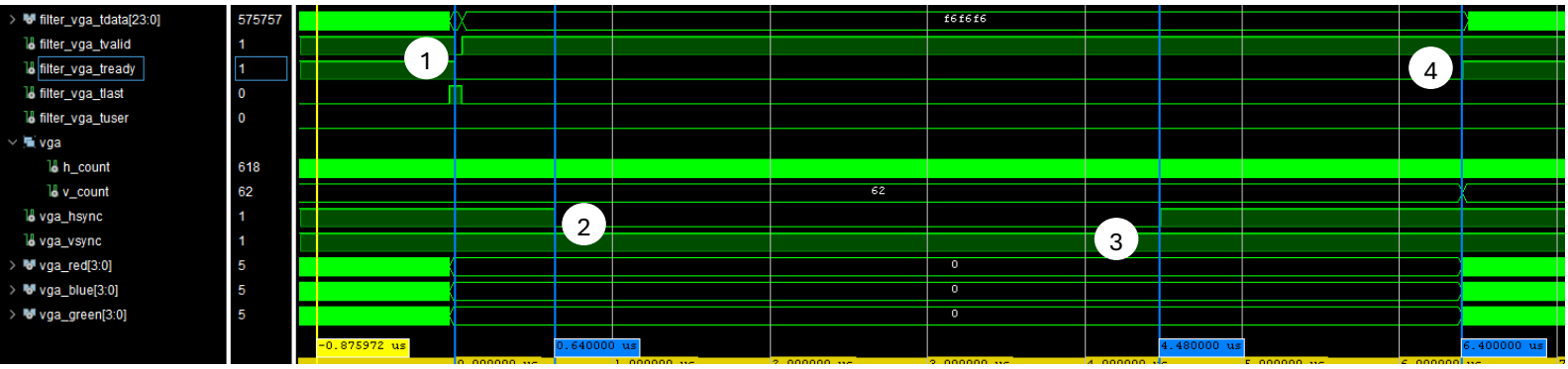
5.1.3. Communication axi stream autour du filtre de gauss



1. Le filtre gauss est prêt à recevoir une donnée axi4s_grey_gauss_tready =1.
2. Le filtre gauss lis la première connée valide du filtre gris axi4s_grey_gauss_tvalid =1.
   2.1 Le handshake_in du filtre gauss est OK: La donnée est reçue
   2.2 Le tuser est aussi lu sur le premier pixel de ll frame
3. La première donnée est prête à être envoyée par le filtre gauss un cycle de ligne plus tard
4. Après être passée dans les filtre gauss , la donnée est prête à être envoyé au bloc suivant
   4.1 axi4s_filter_vga_tvalid = 1
   4.2 donnée sont calculées puis transmises 2 coups d'horloge plus tard sur le handshake_out du filtre.
5. La donnée est envoyée au vga sur le même cycle d'horloge
   5.1 : Le handshake_in du filtre gauss est OK: La donnée est envoyée
6. Les données suivantes sont envoyées au vga sans rupture sur la chaine de données VGA

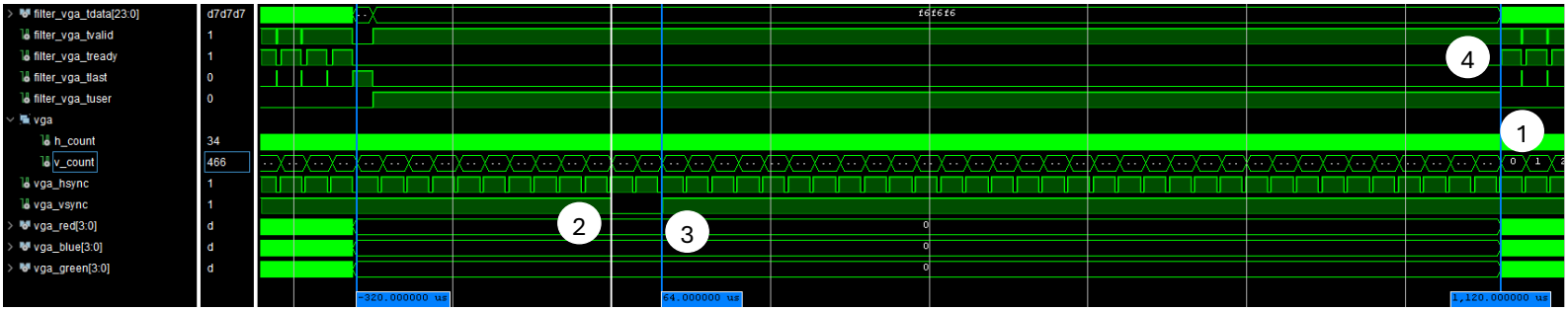
Le Filtre de gauss

5.1.4. Validation des timings hsync



1. Fin de ligne sur le filtre gauss : axi4s_filter_vga_tlast = 1 sur le dernier handshake
2. Le signal hsync passe à 0 0.64us plus tard soit 16* periode(25Mhz)
3. Le signal hsync reste à 0 pendant 96 periode , il repasse à 1 à 4.48us après la fin de ligne
   4.48us = (16+96)* periode(25Mhz)
4. Le protocole VGA dis d'attendre 48 periode supplémentaires avant de lancer la transmission d'un autre ligne
   Le signal tready repasse à 1 à t=6.4us après la fin de ligne pour relancer la transmission d'un autre mot. 6.4us = (16+96+48)* periode(25Mhz)

5.1.5. Validation du timings vsync sur un transmission de frame



1. Au changement de frame v_count et h_count sont remis à zéro
   Les données commencent à être transférées en commençant par la première ligne (v_count = 0)
2. Le signal vsync attend 10 lignes avant de passer à 0 soit 320us = 800*10*periode(25Mhz)après la fin de frame
3. Le signal vsync reste à 0 pendant 2 lignes, soit 64us = 800*2*periode(25Mhz)
4. La transmission est relancée quand 33 cycles de lignes sont passés à t= 1.120us
   (800*(2+33))*periode(25Mhz)

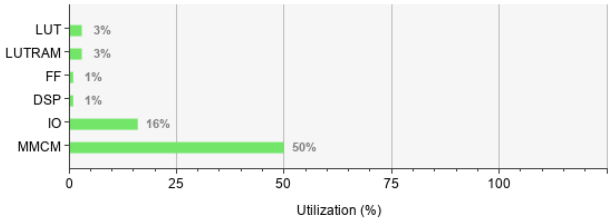
Le Filtre de gauss

5.1. Test sur carte

5.1.1. Synthèse et implémentation : rapport d'utilisation

Name	Slice LUTs (17600)	Slice Registers (35200)	F7 Muxes (8800)	F8 Muxes (4400)	Slice (4400)	LUT as Logic (17600)	LUT as Memory (6000)	DSPs (80)	Bonded IOB (100)	BUFGCTRL (32)	MMCME2_ADV (2)
top_tpg_grey_gauss_vga	534	239	80	32	184	374	160	1	16	2	1
u_axi4s_to_vga (axi4s_to_vga)	33	21	0	0	11	33	0	0	0	0	0
u_clk_wiz_0 (clk_wiz_0)	0	0	0	0	0	0	0	0	0	2	1
u_filter_gauss (filter_gauss)	344	130	80	32	106	184	160	0	0	0	0
u_filter_grey (filter_grey)	17	13	0	0	7	17	0	1	0	0	0
u_pattern_generator (pattern_generator)	141	75	0	0	63	141	0	0	0	0	0

Resource	Utilization	Available	Utilization %
LUT	545	17600	3.10
LUTRAM	160	6000	2.67
FF	239	35200	0.68
DSP	1	80	1.25
IO	16	100	16.00
MMCM	1	2	50.00



Ref Name	Used	Functional Category
FDCE	229	Flop & Latch
LUT2	173	LUT
RAMS64E	160	Distributed Memory
LUT6	118	LUT
MUXF7	80	MuxFx
LUT4	76	LUT
LUT3	66	LUT
LUT5	49	LUT
CARRY4	49	CarryLogic
MUXF8	32	MuxFx
LUT1	32	LUT
OBUF	14	IO
FDPE	10	Flop & Latch
IBUF	2	IO
BUFG	2	Clock
MMCME2_ADV	1	Clock
DSP48E1	1	Block Arithmetic

Toujours le même nombre de registre utilisés pour axi4s_to_vga et pattern generator.
Pour l'implémentation du filtre gris : 13 registres pour les signaux axis
Pour l'implémentation du filtre gauss :

130 registres pour la matrice de calcul, les compteurs, les bordures et les signaux axi
160 LUT mémoire pour les 2 lines buffers

5.1.1. Synthèse et implémentation : rapport de timings

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	25,972 ns	Worst Hold Slack (WHS):	0,042 ns	Worst Pulse Width Slack (WPWS):	2,000 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	2020	Total Number of Endpoints:	2020	Total Number of Endpoints:	406

All user specified timing constraints are met.

Les contraintes de timing sont respectées avec une horloge à 25Mhz.

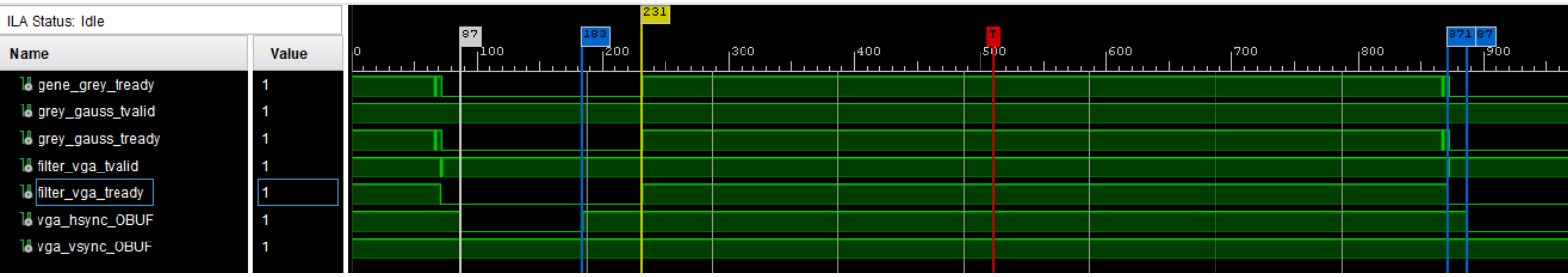
Summary

Name	Path 1
Slack	25.972ns
Source	u_pattern_generator/box_x_reg_reg[3]/C (rising edge-triggered cell FDCE clocked by clk_out1_clk_wiz_0 {rise@0.000ns fall@20.000ns period=40.000ns})
Destination	u_filter_grey/multOp/A[9] (rising edge-triggered cell DSP48E1 clocked by clk_out1_clk_wiz_0 {rise@0.000ns fall@20.000ns period=40.000ns})
Path Group	clk_out1_clk_wiz_0
Path Type	Setup (Max at Slow Process Corner)
Requirement	40.000ns (clk_out1_clk_wiz_0 rise@40.000ns - clk_out1_clk_wiz_0 rise@0.000ns)
Data Path Delay	9.899ns (logic 3.918ns (39.579%) route 5.981ns (60.421%))
Logic Levels	13 (CARRY4=7 LUT1=1 LUT2=1 LUT3=1 LUT4=2 LUT6=1)
Clock Path Skew	-0.041ns
Clock Un...rtainty	0.160ns

Le chemin critique est un chemin entre les données du pattern generator et la logique de calcul du filtre gris

Le Filtre de gauss

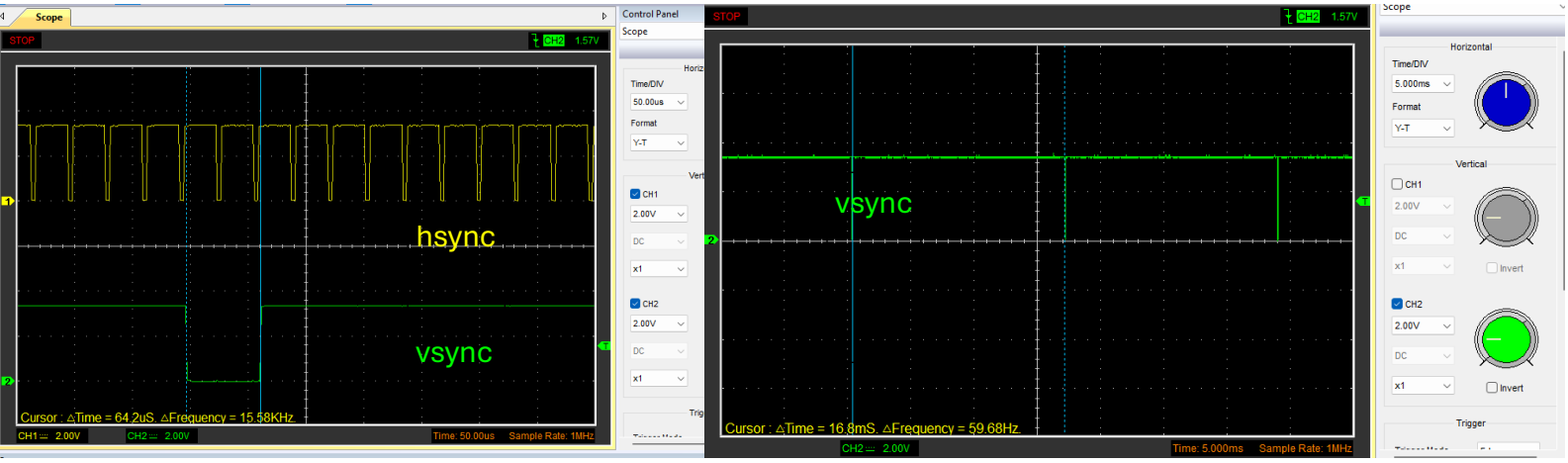
5.1.2. Mesure ILA



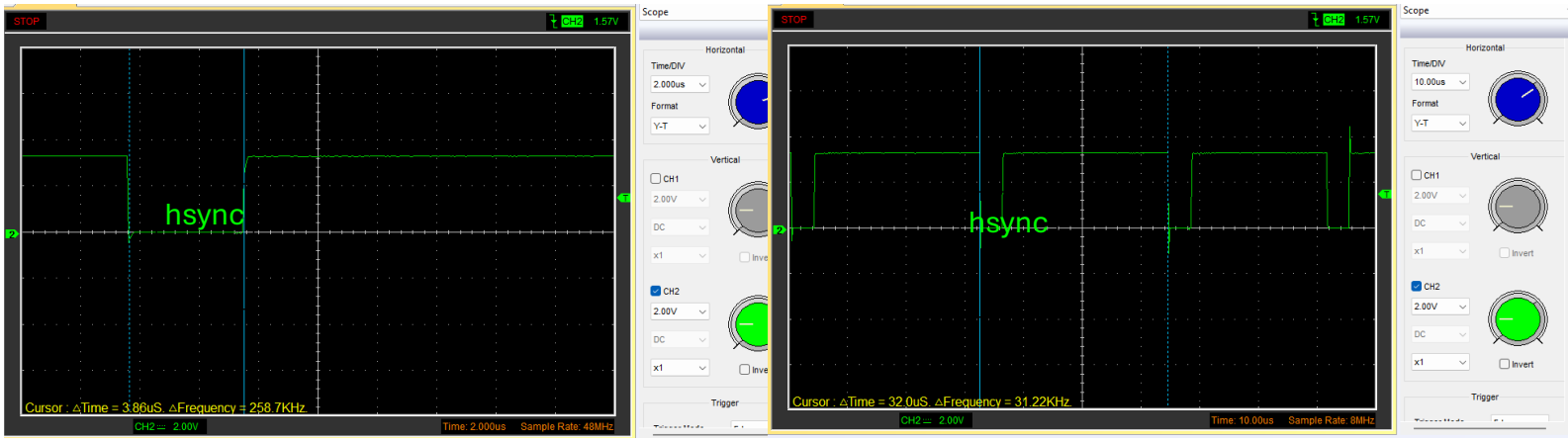
tready reste à 1 sur un ligne pendant 871-231= 640*periode(clk)
un cycle hsync dure 887-87 = 800* periode(clk)
hsync est à 0 pendant 183-87 = 96*periode(clk)
tready passe à 1 231-183 = 48*periode(clk) après que hsync soit repassée à 1

Sur une échantillonnage de 1024 mesures, il n'est pas possible de capturer le passage à zéro de Vsync qui dure 1600 cycles d'horloge

5.1.3. Mesure sur oscilloscope



Le signal vsync passe à 0 toutes les 16.8ms = 800*525* periode(25Mhz)
le signal vsync reste à 0 pendant 2 cycles hsync soit 64us= 800*2*periode(25Mhz)

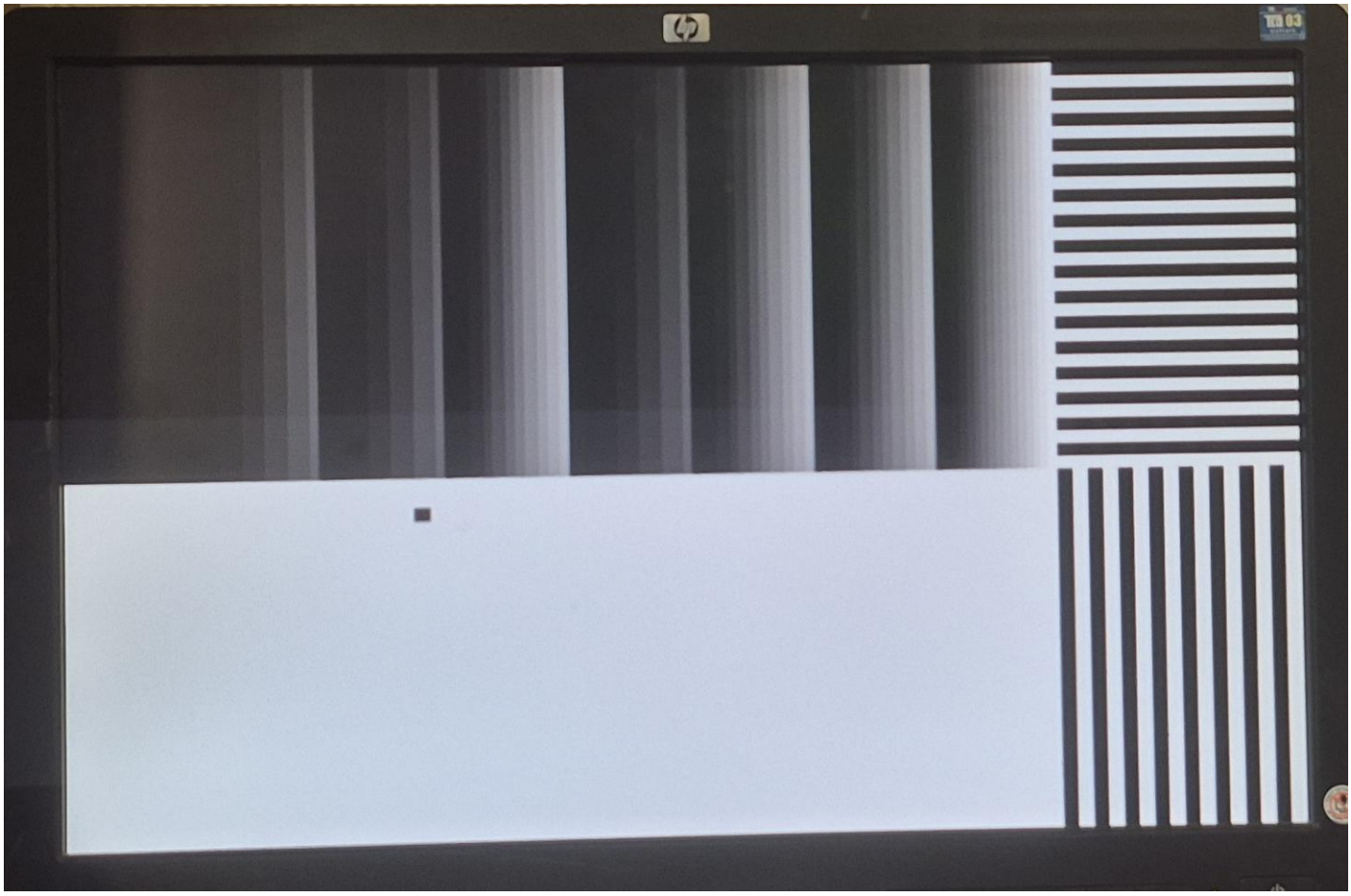


hsync est à 0 pendant 3.83 us ~ 96*periode(25Mhz)
hsync à une période de 32us = 800*periode(clk)

5.1. Démo

L'image est affichée sur l'écran , on vérifier que celle-ci et stable et ne présenta visuellement de décalage de ligne ou de pixel ou de colonne

Le Filtre de gauss



Le pattern couleur du départ a bien été converti en niveau de gris puis floutté par le filtre de gauss
L'image ne présente pas de décalage de pixel visible.
Le point noir dans le cadre en bas à gauche se déplace dans ce cadre.

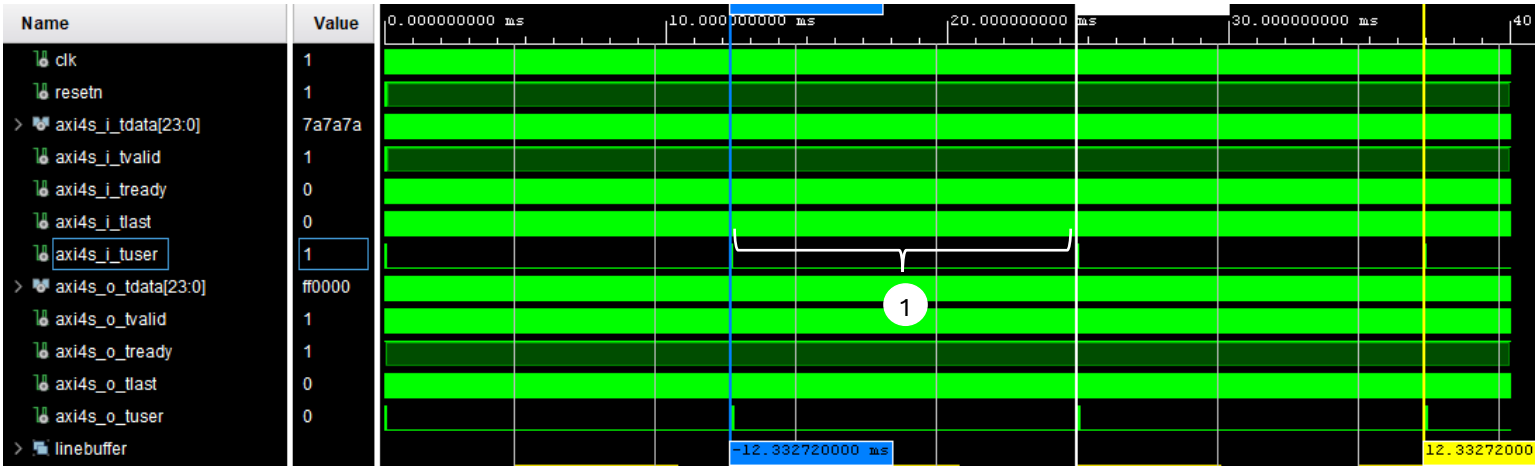
6. Le Filtre harris

6.1. Validation du bloc - simulation

6.1.1. Test bench

Le test bench utilisé consiste à utiliser 3 générateurs de trame axis4s fournit (axi4sdriver. vhd)
axi4sdriver fournit une trame axi4s à partir d'un fichier texte en entrée.
Les 3 fichiers texte doivent contenir un table de 640*480 mots de 8bit en décimal
Les 3 fichiers d'entrée sont identiques et comportent le résultat du filtrage gauss.
Afin de faciliter la validation du transfert des données, les fichiers texte utilisés en entrée ont été remplis de mots de 8 bits générés de façon aléatoire et de sorte que 2 nombres consécutifs ne soient pas identiques

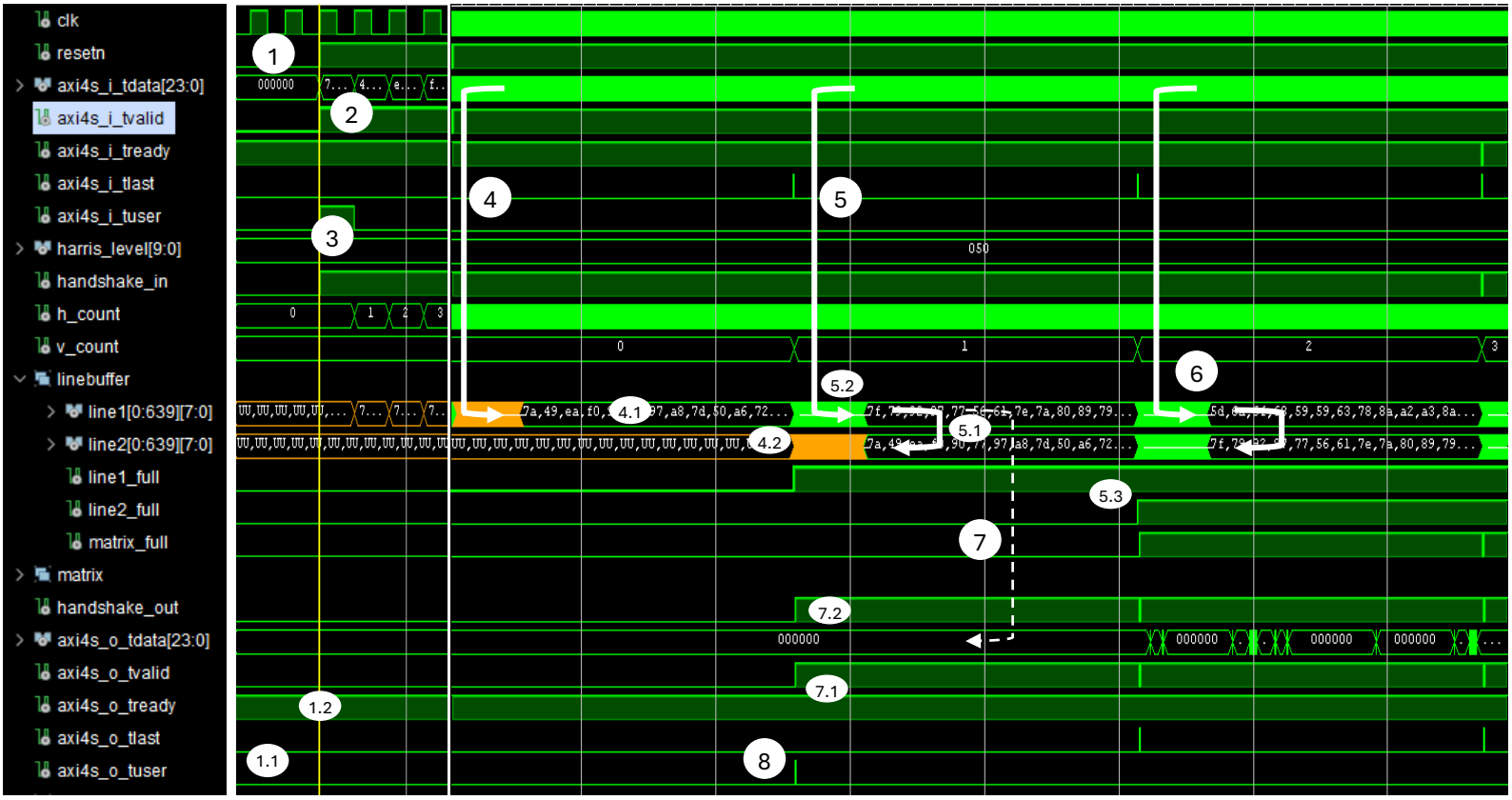
6.1.2. Validation fonctionnelle



1. Le filtre harris transmet les données reçues sur l'axi stream d'entrée sur l'axis stream de sortie après avoir calculé la valeur harris à transmettre
   Chaque frame est délimitée par le signal tuser .
   Une frame est émise toutes les 12.33272ms = (640*480) + 1118*periode(25Mhz)

Les 1118 périodes supplementaires correspondent au delais introduits par l'architecture du filtre de harris.: 640 cycles pour le delai d'attente de remplissage du line buffer 1
478 cycle pour le transfert du dernier pixel de chaque lignes sauf pour la première et dernière ligne de la frame.

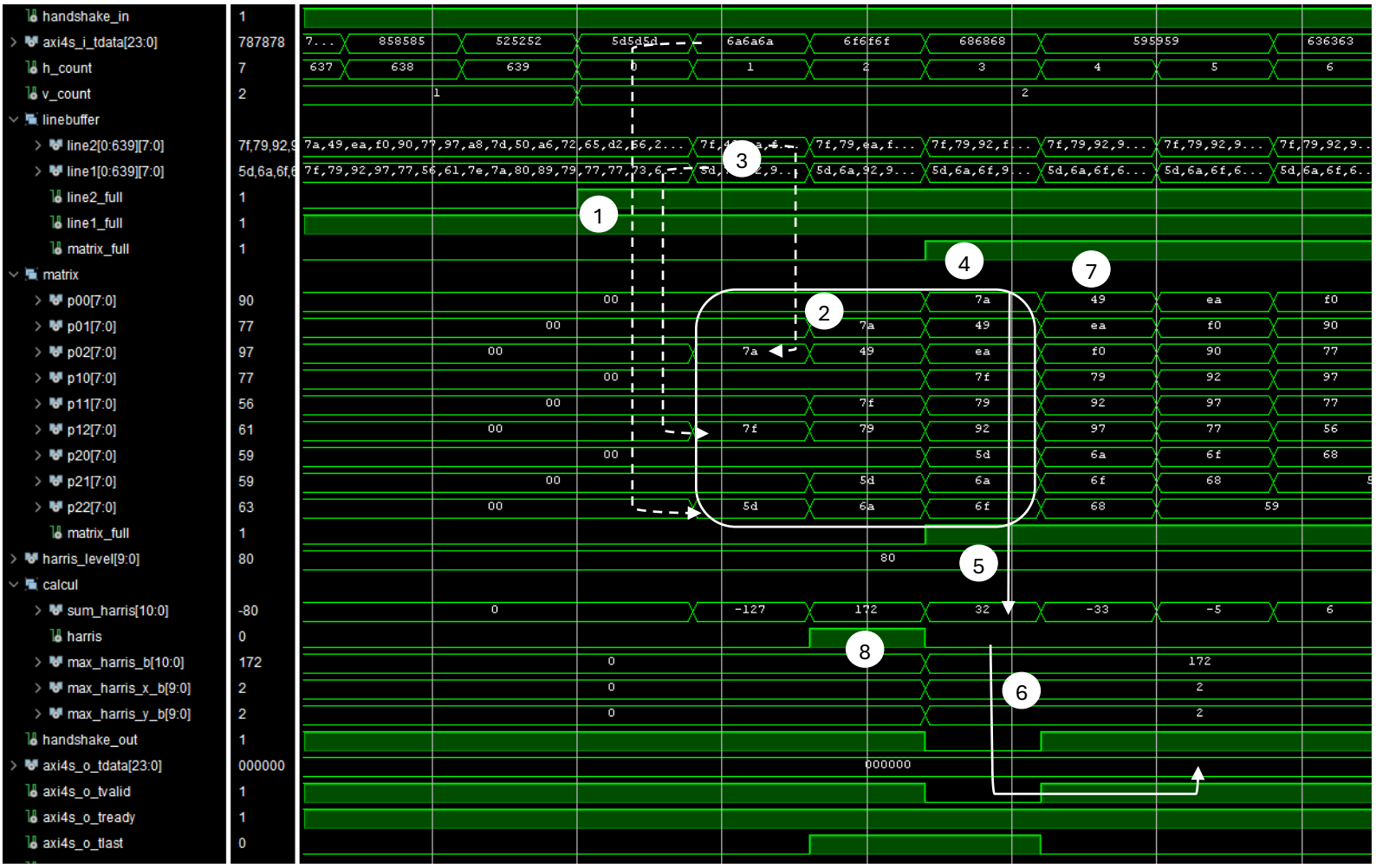
6.1.3. Démarrage après reset et premier transfert de données



zoom reset

1. Au reset:
   1.1 Les signaux de sortie de contrôle axi4s_o_... sont à zéro: tvalid, tuser, tlast.
   1.2 Le signal tready est à 1 pour indiquer que l'axi slave est prêt pour recevoir de la donnée.
2. La transmission commence après le lever du reset quand le générateur de flux axis_driver commence à envoyer des données
   - axi4s_i_tvalid passe à 1
   - Le filtre était prêt à recevoir axi4s_i_tready = 1
   - Le handshake_in = 1: le filtre peut recevoir des données
3. La première donnée est identifiée par axi4s_i_tuser = 1
4. La première ligne est lue et stockée dans le line1 buffer
   4.1le line1 buffer se rempli
   4.2 jusqu'à ce qu'il soit plein à la fin de la première ligne lue
5. La seconde ligne est lue et stockée dans le line1 buffer
   5.1 Le line1 buffer 1 est transféré dans le line2buffer
   5.2 Le line1buffer se rempli avec les données lues sur axi4s_i_data
   5.3 A la fin de lecture de la seconde ligne de données les deux line buffers sont pleins
6. Ce remplissage des buffers continue sur les autres lignes lues
7. Dès que le premier line1 buffer est plein, on peut commencer à transférer les données en sortie du filtre
   7.1 axi4s_o_tvalid passe à 1
   le bloc en sortie était prêt à recevoir axi4s_o_tready = 1
   7.2 Le handshake_out = 1: le filtre peut transférer les données
   La première lignes est envoyée en sortie sans opérations de calcul
   Pour le filtre harris , il est envoyé des 0 sur toute la bordure
8. Le signal axi4s_o_tuser passe à 1 sur le premier pixel de la frame transmis

6.1.4. Calcul du filtrage : line buffer et matrice



1. Dès que les 2 lines buffers sont plein , la matrice de calcul peut commencer à se remplir
2. La matrice se remplie avec les contenue de line1, line2 et i_data
3. En même temps les lines buffers continuent de se remplir aussi
4. Après 3 coup d'horloge , La matrice est pleine est pleine
5. Les calculs sont exécutés et pris en compte dans le transfert des données quand la matrice est pleine.
6. La valeur calculée est transmise sur le bus de data 2 coup d'horloge plus tard suivant le protocole des registre du filtre gauss. Ici on ne voit pas de changements car la sortie est à 0 et reste à zéro
7. .Le même calcul est exécuté sur les pixels suivants
8. Cas particuliers: ON voit ici une sortie du calcul harris qui passe à 1 , mais on ne voit aucune transmission à 1 sur la lignes de sortie o_data. Ce pixel correspond au premier pixel de la deuxième ligne . Comme toute la bordure de sortie du filtre harris est envoyé avec un 0 , on envoi donc un 0 sur o_data pour ce pixel

6.1.5. Calcul du filtre harris

La transimission des données est décalé d'une ligne complète . Il est difficile de lereprésenter sur un chronogramme dans word .

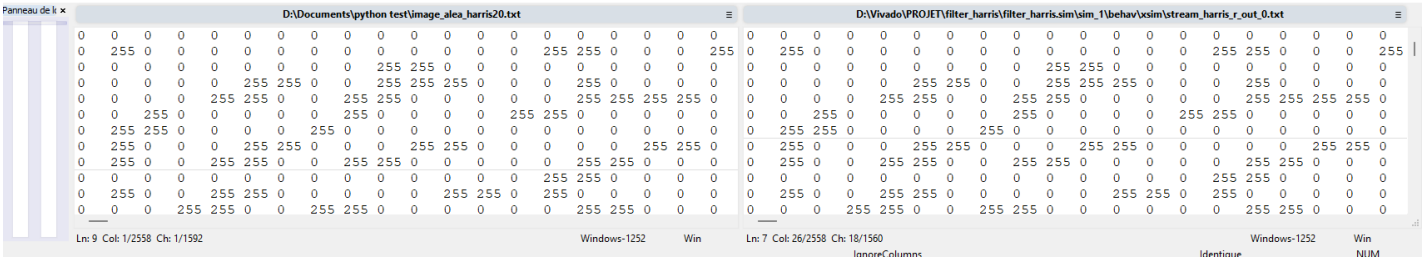
Les checks de debut et fin de ligne ainsi que les checks sur les premières et dernières lignes sont des cas particuliers car sur ces pixels la sortie est forcée à 0.

Afin de vérifier que la transmission s'est bien effectuée sur tous les pixel de la frame, on compare le ficher résultat de la simulation avec le fichier résultat théorique calculé.

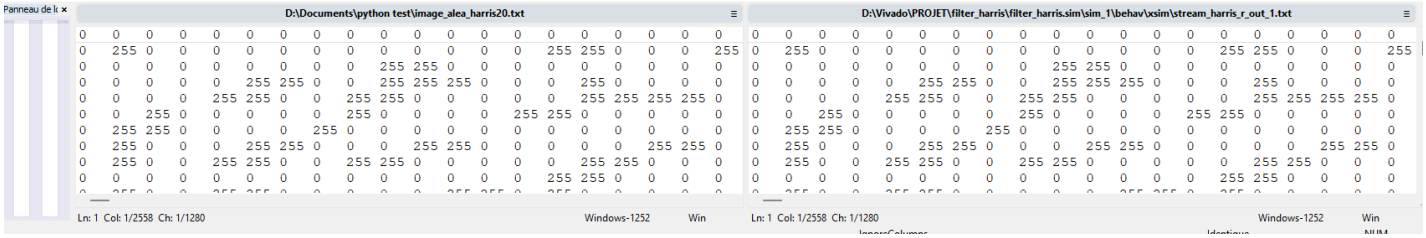
L'outil winmerge permet de comparer ces fichiers.

La comparaison se fait sur les 2 premières frames générées par les test bench

Le Filtre harris



Première frame générée sans erreur. S'il y avait une erreur elle serait affichée dans la colonne de gauche

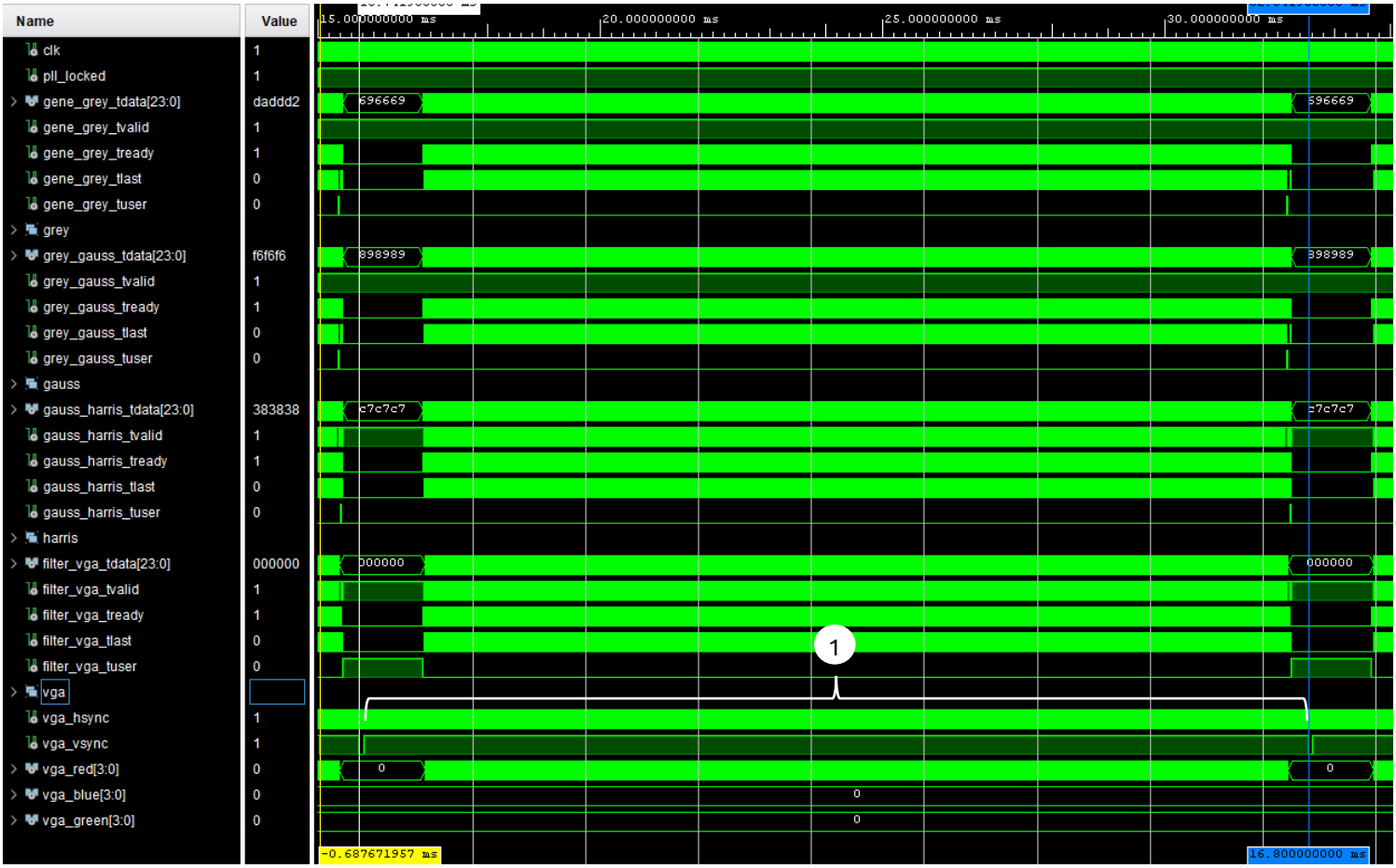


Deuxième frame générée sans erreur.

6.2. Validation du filtre de gauss – Intégration dans le top

Les blocs présents pour cette simulation sont la pll, le pattern generator, le filtre gris, le filtre de gauss, le filtre harris et le convertisseur axi4s_to_vga.

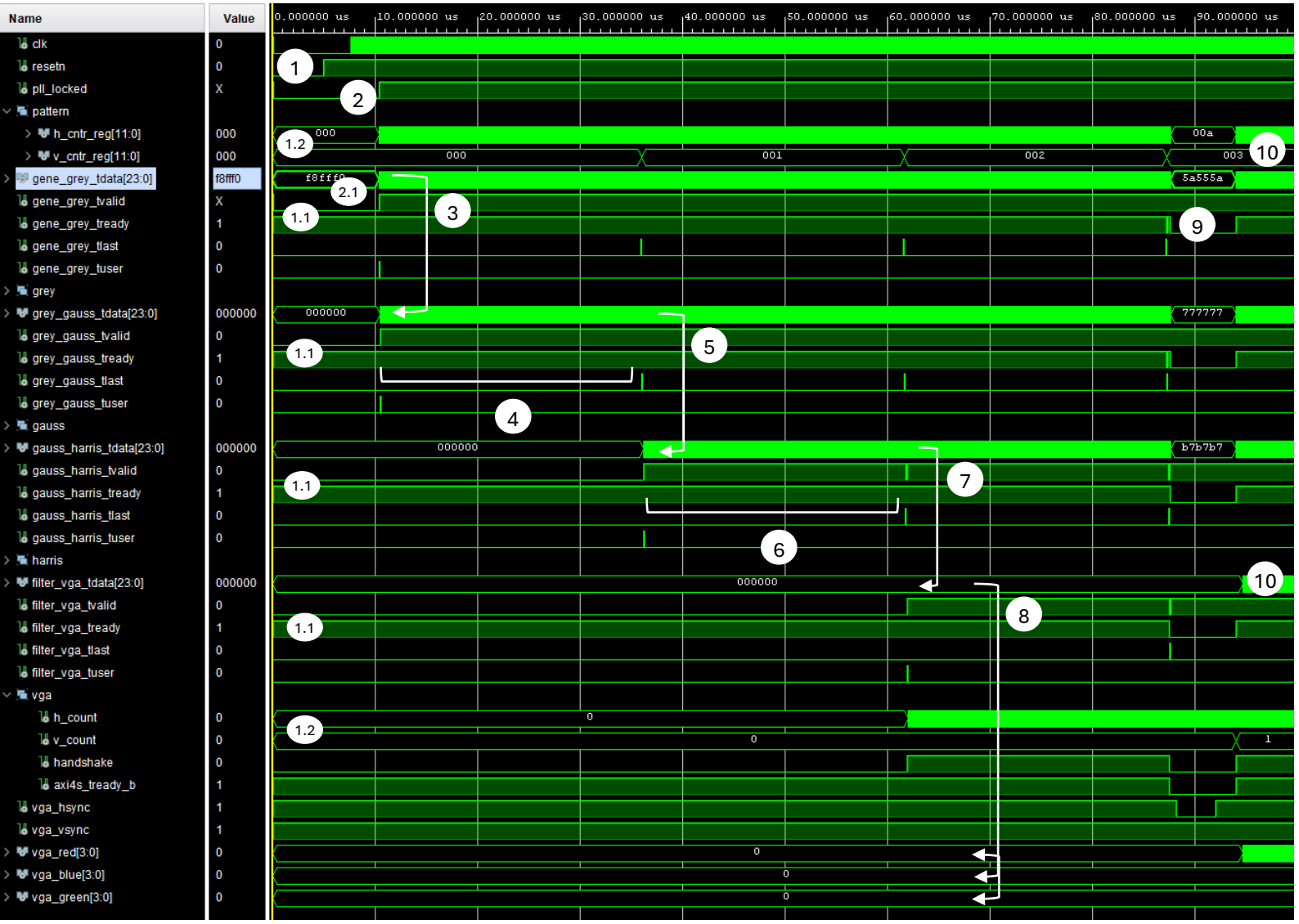
6.2.1. Validation fonctionnelle



1. Le pattern generator crée une frame transférée en axi stream sur le filtre gris, puis sur le filtre gauss puis sur le filtre harris
   Le signal filtré en envoyé sur le convertisseur axi4s_to_vga
   Cette communication respecte le protocole VGA
   Une frame est émise toutes les 16.8ms = 800*525*periode(25Mhz)

Le Filtre harris

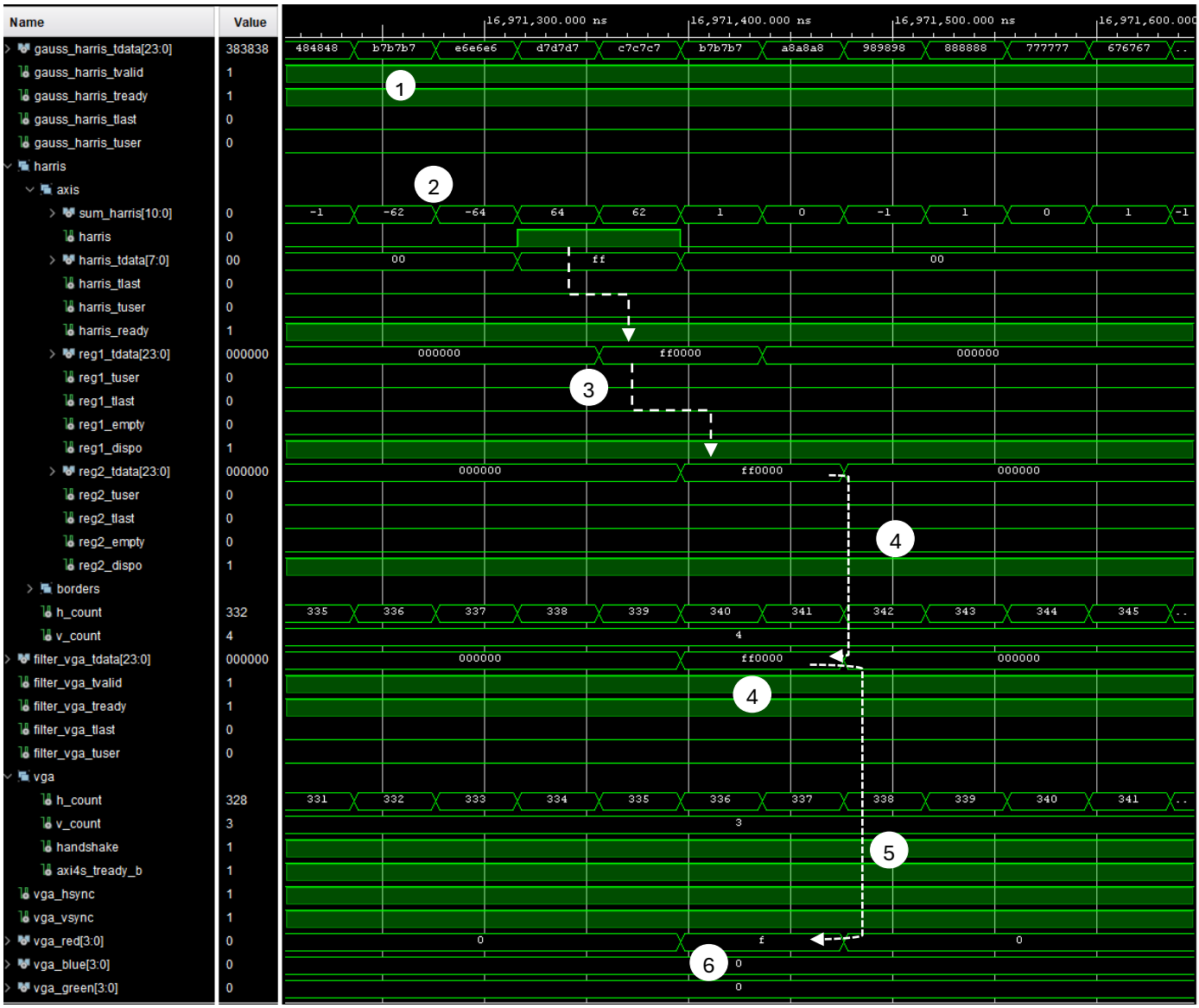
6.2.2. Démarrage après reset



1. Au reset, le signaux de sortie de contrôle axi stream sont à zéro: tvalid, tuser, tlast.
   1.1 Les signaux tready est à 1 pour indiquer que les axi slave seront prêt pour recevoir de la donnée au démarrage.
   1.2 Les compteurs du pattern et du vga sont à 0.
2. Le pattern generator est mis en marche quand la pll a démarré sur pll_locked
   2.1Le tvalid suit pattern_en et indique que les données sont prêtes à être envoyées
3. Les données générées par le pattern sont envoyées au filtre gris sur le handshake gene_grey
   gene_grey_tready = 1 et gene_grey_tvalid = 1
4. Le filtre de gauss commence à fournir des données au vga un cycle de ligne plus tard
5. Les données générées par le filtre gris sont envoyées au filtre gauss sur le handshake grey_gauss
   grey_gauss_tready = 1 et grey_gauss_tvalid= 1
6. Le filtre harris commence à fournir des données au vga un cycle de ligne plus tard
7. Les données générées par le filtre gauss sont envoyées au filtre harris sur le handshake gauss harris
   gauss_harris_tready = 1 et gauss_harris_tvalid
8. Les données générées par le filtre harris sont envoyées au vga sur le handshake filter_vga
   filter_vga_tready = 1 et filter_vga_tvalid= 1
9. La première ligne est transmise, le pattern generator se met en pause en attendant la fin du protocole vga gene_grey_tready = grey_gauss_tready = filter_vga_tready 0
10. Le protocole vga est terminé, on passe à l'affichage de la deuxième ligne, la pattern generator et toute la ligne de filtrage redémarre pour le calcul de la deuxième ligne

Le Filtre harris

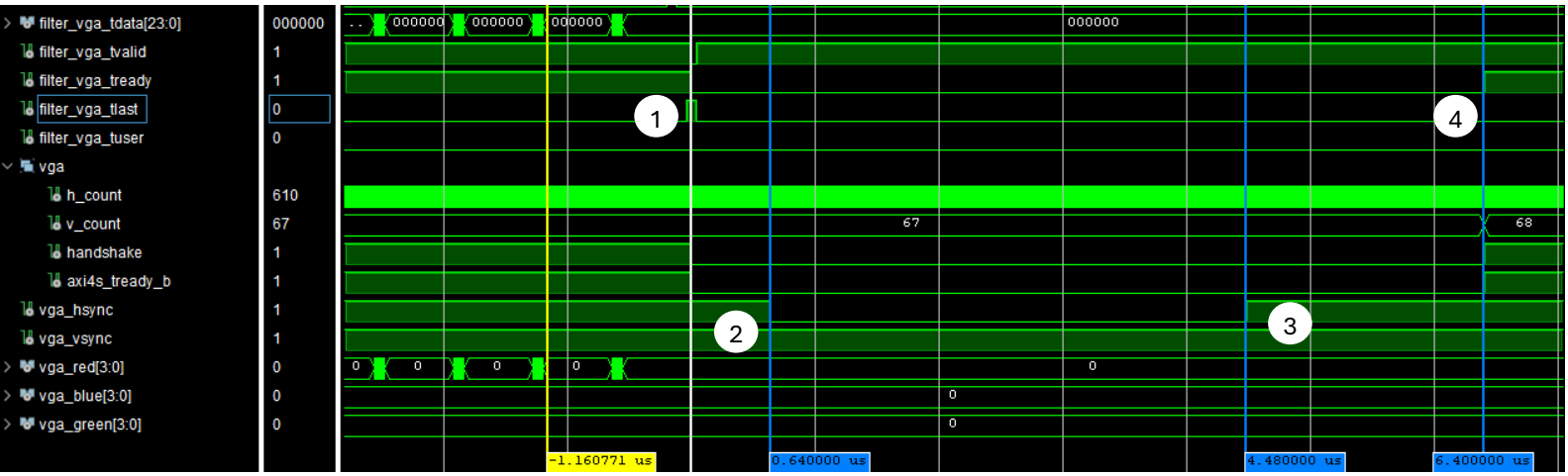
6.2.3. Communication axi stream autour du filtre harris



1. L' handshake d'entrée du filtre harris est OK axi4s_gauss_harris_tready =1 et . axi4s_gauss_harris_tvalid =1
2. Les données sont reçues et traitées sur le filtre harris
3. Les données calculées sont envoyées sur les registres du filtre harris en attente d'être envoyées si un handshake de sortie a lieu
4. Après être passée dans les filtre gauss , la donnée est prête à être envoyé au bloc suivant
   4.1 axi4s_filter_vga_tvalid = 1
   4.2 donnée sont calculées puis transmises 2 coups d'horloge plus tard sur le handshake_out du filtre.
5. La donnée est envoyée au vga sur le même cycle d'horloge
   5.1 : Le handshake_in du filtre harris est OK: La donnée est envoyée
6. Le contour détecté sera allumé en rouge sur l'ecran

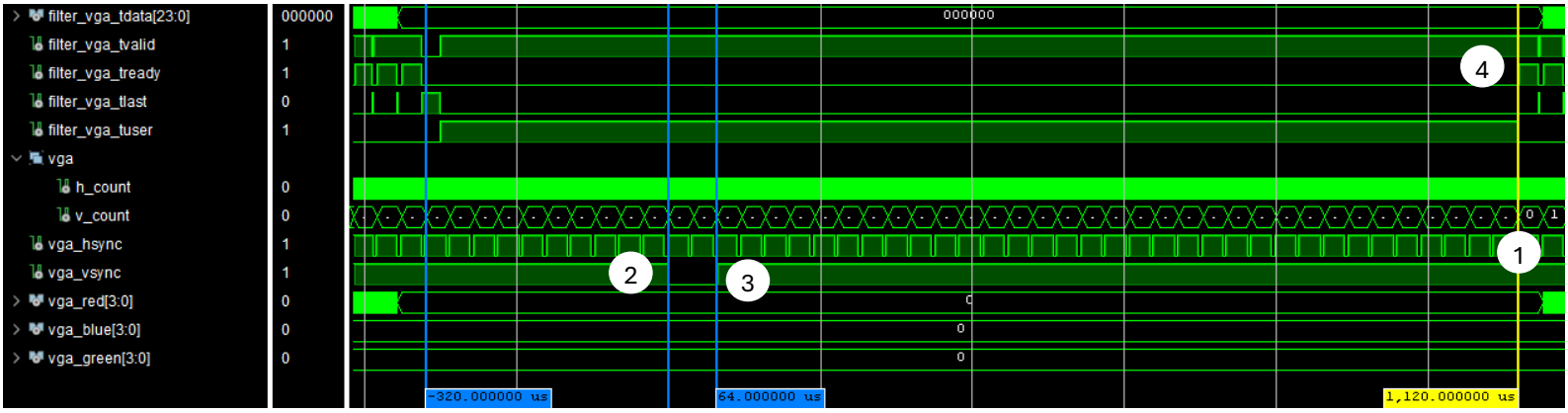
Le Filtre harris

6.2.4. Validation des timings hsync



1. Fin de ligne sur le filtre harris : axi4s_filter_vga_tlast = 1 sur le dernier handshake
2. Le signal hsync passe à 0 0.64us plus tard soit 16* periode(25Mhz)
3. Le signal hsync reste à 0 pendant 96 periode , il repasse à 1 à 4.48us après la fin de ligne
   4.48us = (16+96)* periode(25Mhz)
4. Le protocole VGA dis d'attendre 48 periode supplémentaires avant de lancer la transmission d'un autre ligne
   Le signal tready repasse à 1 à t=6.4us après la fin de ligne pour relancer la transmission d'un autre mot. 6.4us = (16+96+48)* periode(25Mhz)

6.2.5. Validation du timings vsync sur un transmission de frame



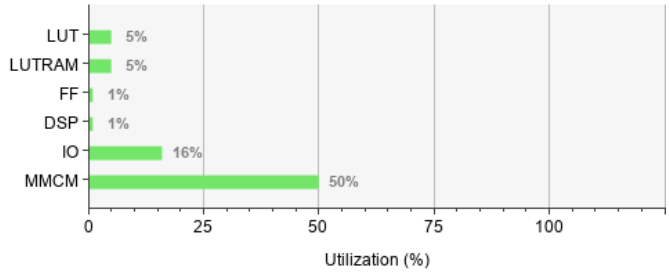
1. Au changement de frame v_count et h_count sont remis à zéro
   Les données commencent à être transférées en commençant par la première ligne (v_count = 0)
2. Le signal vsync attend 10 lignes avant de passer à 0 soit 320us = 800*10*periode(25Mhz)après la fin de frame
3. Le signal vsync reste à 0 pendant 2 lignes, soit 64us = 800*2*periode(25Mhz)
4. La transmission est relancée quand 33 cycles de lignes sont passés à t= 1.120us
   (800*(2+33))*periode(25Mhz)

6.3. Test sur carte

6.3.1. Synthèse et implémentation : rapport d'utilisation

Name	Slice LUTs (17600)	Slice Registers (35200)	F7 Muxes (8800)	F8 Muxes (4400)	DSPs (80)	Bonded IOB (100)	BUFGCTRL (32)	MMCME2_ADV (2)
top_tpg_grey_gauss_har_vga	836	355	160	64	1	16	2	1
u_axi4s_to_vga (axi4s_to_vga)	34	21	0	0	0	0	0	0
u_clk_wiz_0 (clk_wiz_0)	0	0	0	0	0	0	2	1
u_filter_gauss (filter_gauss)	337	154	80	32	0	0	0	0
u_filter_grey (filter_grey)	19	13	0	0	1	0	0	0
u_filter_harris (filter_harris)	302	92	80	32	0	0	0	0
u_pattern_generator (pattern_generator)	143	75	0	0	0	0	0	0

Resource	Utilization	Available	Utilization %
LUT	836	17600	4.75
LUTRAM	320	6000	5.33
FF	355	35200	1.01
DSP	1	80	1.25
IO	16	100	16.00
MMCM	1	2	50.00



Primitives

Ref Name	Used	Functional Category
FDCE	343	Flop & Latch
RAMS64E	320	Distributed Memory
LUT2	239	LUT
MUXF7	160	MuxFx
LUT6	146	LUT
LUT4	124	LUT
LUT3	87	LUT
LUT5	78	LUT
MUXF8	64	MuxFx
CARRY4	57	CarryLogic
LUT1	38	LUT
OBUF	14	IO
FDPE	12	Flop & Latch
IBUF	2	IO
BUFG	2	Clock
MMCME2_ADV	1	Clock
DSP48E1	1	Block Arithmetic

Toujours le même nombre de registre utilisés pour axi4s_to_vga et pattern generator.
Pour l'implémentation du filtre gris : 13 registres pour les signaux axis
Pour l'implémentation du filtre gauss :

154 registres pour la matrice de calcul, les compteurs, les bordures et les signaux axi
160 LUT mémoire pour les 2 lines buffers

Pour l'implémentation du filtre harris :

92 registres pour la matrice de calcul, les compteurs, les bordures et les signaux axi
160 LUT mémoire pour les 2 lines buffers

6.3.2. Synthèse et implémentation : rapport de timings

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	26,570 ns	Worst Hold Slack (WHS):	0,083 ns	Worst Pulse Width Slack (WPWS):	2,000 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	3811	Total Number of Endpoints:	3811	Total Number of Endpoints:	682

All user specified timing constraints are met.

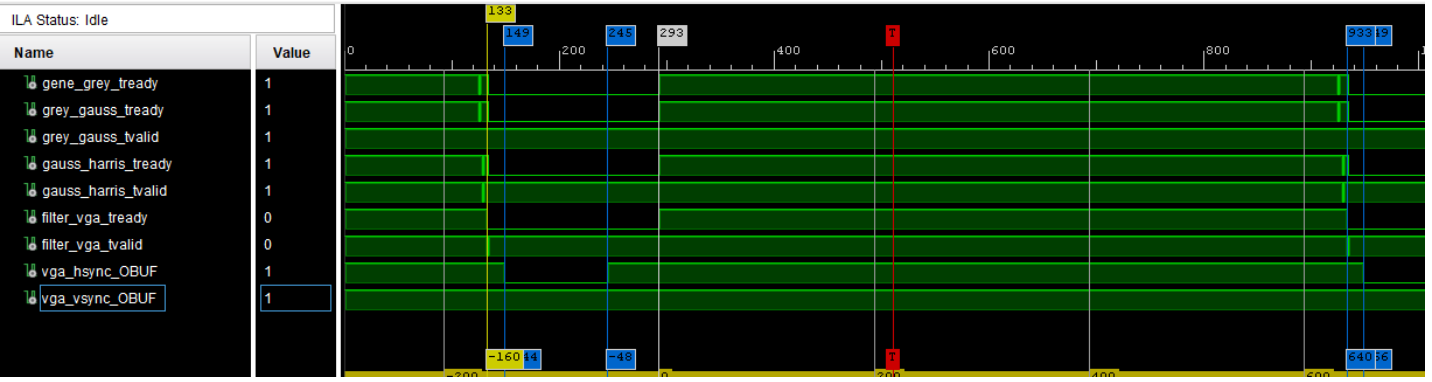
Les contraintes de timing sont respectées avec une horloge à 25Mhz.

Le Filtre harris

Summary	
Name	Path 1
Slack	26.570ns
Source	u_pattern_generator/box_y_reg_reg[5]/C (rising edge-triggered cell FDCE clocked by clk_out1_clk_wiz_0 {rise@0.000ns fall@20.000ns period=40.000ns})
Destination	u_filter_grey/multOp/A[5] (rising edge-triggered cell DSP48E1 clocked by clk_out1_clk_wiz_0 {rise@0.000ns fall@20.000ns period=40.000ns})
Path Group	clk_out1_clk_wiz_0
Path Type	Setup (Max at Slow Process Corner)
Requirement	40.000ns (clk_out1_clk_wiz_0 rise@40.000ns - clk_out1_clk_wiz_0 rise@0.000ns)
Data Path Delay	9.224ns (logic 3.810ns (41.307%) route 5.414ns (58.693%))
Logic Levels	10 (CARRY4=5 LUT2=1 LUT3=1 LUT4=2 LUT6=1)
Clock Path Skew	-0.145ns
Clock Un...rtainty	0.160ns
Source Clock Path	

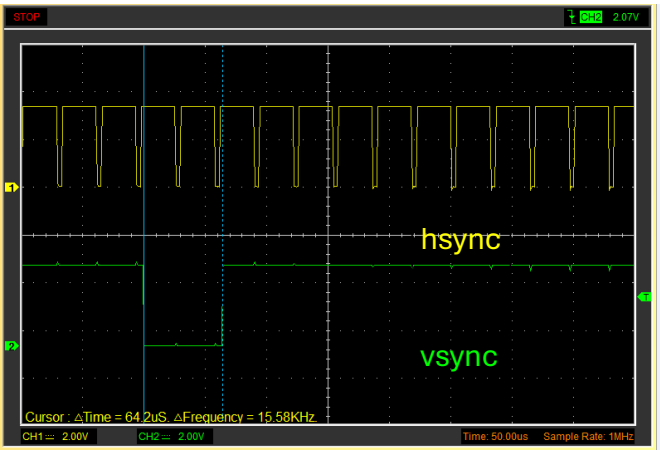
Toujours le même chemin critique entre les données du pattern generator et la logique de calcul du filtre gris

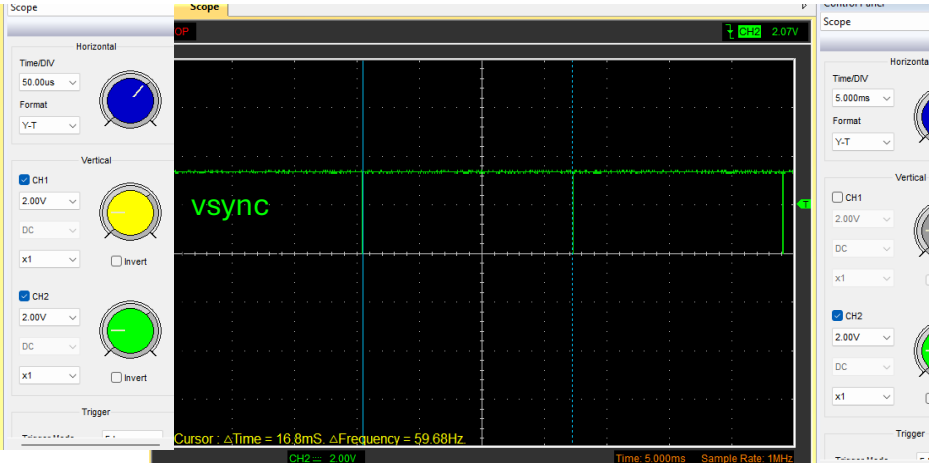
6.3.3. Mesure ILA



tready reste à 1 sur un ligne pendant 933-293= 640*periode(clk)
un cycle hsync dure 949-149 = 800* periode(clk)
hsync est à 0 pendant 245-149 = 96*periode(clk)
tready passe à 1 293-245 = 48*periode(clk) après que hsync soit repassée à 1
hsync passe à 0 149-133 = 16 cycles après que tready =0

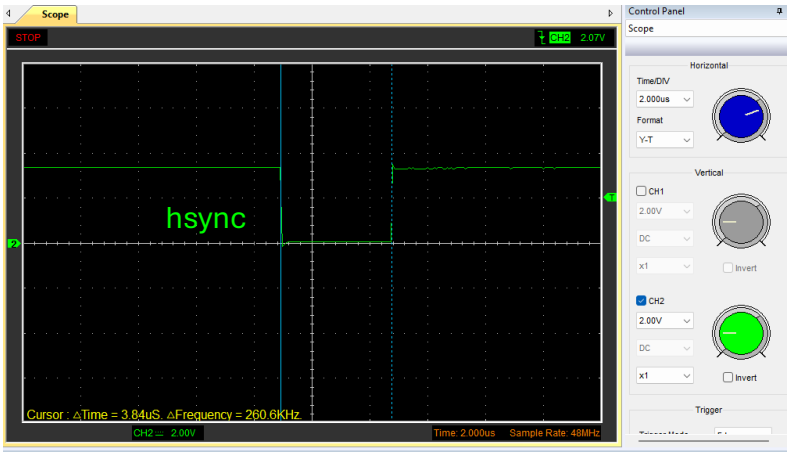
6.3.4. Mesure sur oscilloscope

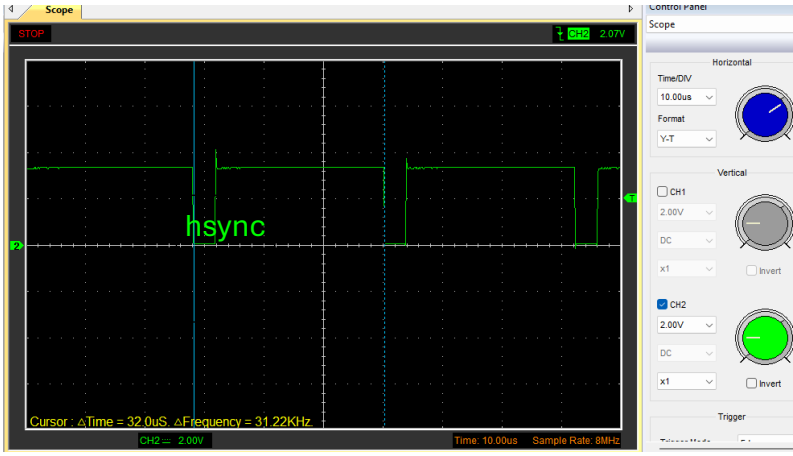




Le signal vsync passe à 0 toutes les 16.8ms = 800*525* periode(25Mhz)
le signal vsync reste à 0 pendant 2 cycles hsync soit 64us= 800*2*periode(25Mhz)

Le Filtre harris





hsync est à 0 pendant 3.83 us ~ 96*periode(25Mhz)
hsync à une période de 32us = 800*periode(clk)

6.4. Démo

L'image est affichée sur l'écran , on vérifier que celle-ci et stable et ne présenta visuellement de décalage de ligne ou de pixel ou de colonne

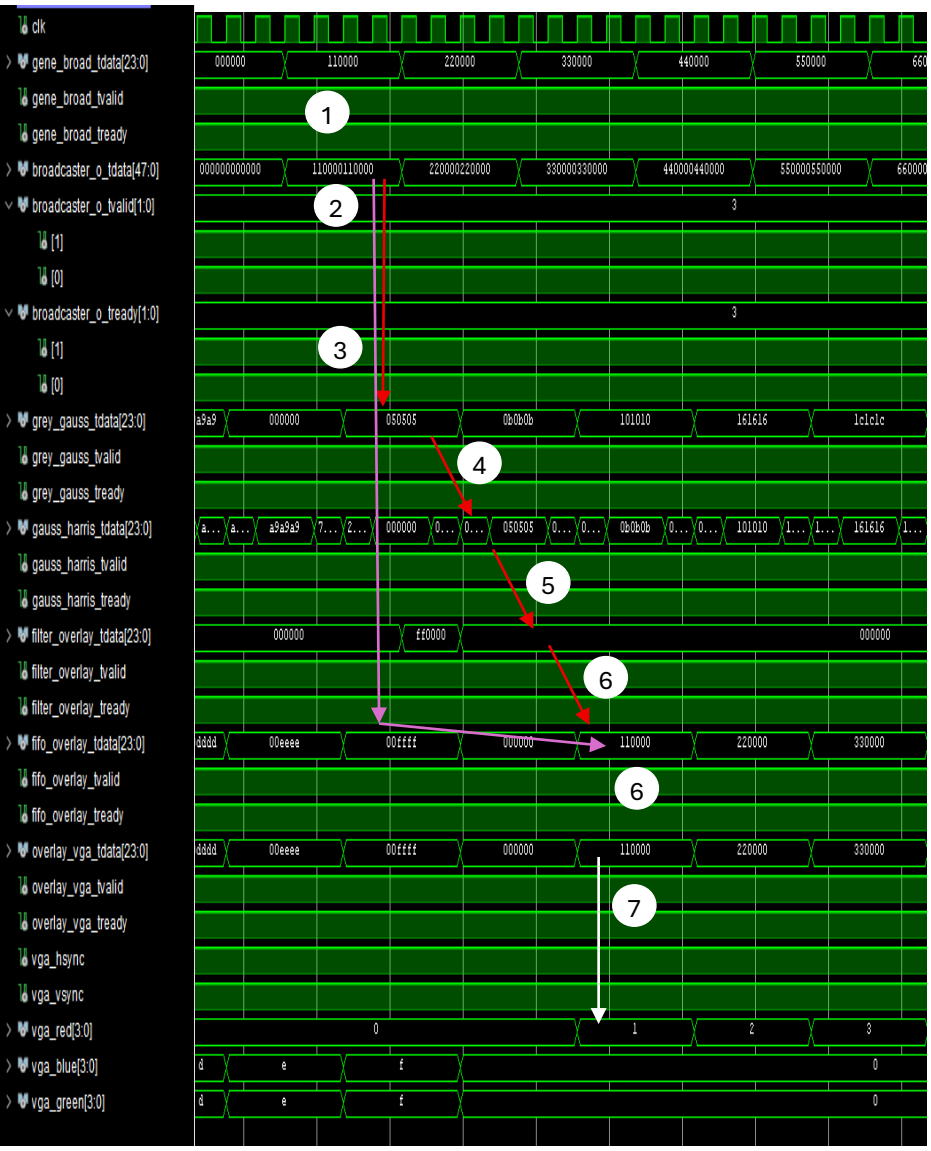
Le pattern couleur du départ a bien été filtré, on ne voit plus que les contours du pattern
L'image ne présente pas de décalage de pixel visible.
Le point noir a bien été détecté , son contour se déplace dans le cadre en bas à gauche.

7. Ligne complète

7.1. Validation de la ligne complète en simulation

Les blocs présent pour cette validation sont la pll, le pattern generator, le filtre gris, le filtre de gauss, le filtre harris ; le broadcaster, la fifo, l'overlay et le convertisseur axi4s_to_vga

7.1.1. Protocole axi stream sur la chaine



1. Sur le handshake gene_broad:
La data est transmise directement sur le broadcaster

2. La data est doublée dans le broadcaster pour être envoyée sur le premier filtre gris et sur la fifo

3. Sur le handshake broad_grey et broad_fifo, la data est transmise en entrée du filtre gris pur être traitée et elle est aussi envoyé dans la queue de la fifo

4. Sur le handshake grey_gauss:
La data est transmise au filtre gauss 2 coup d'horloge plus tard

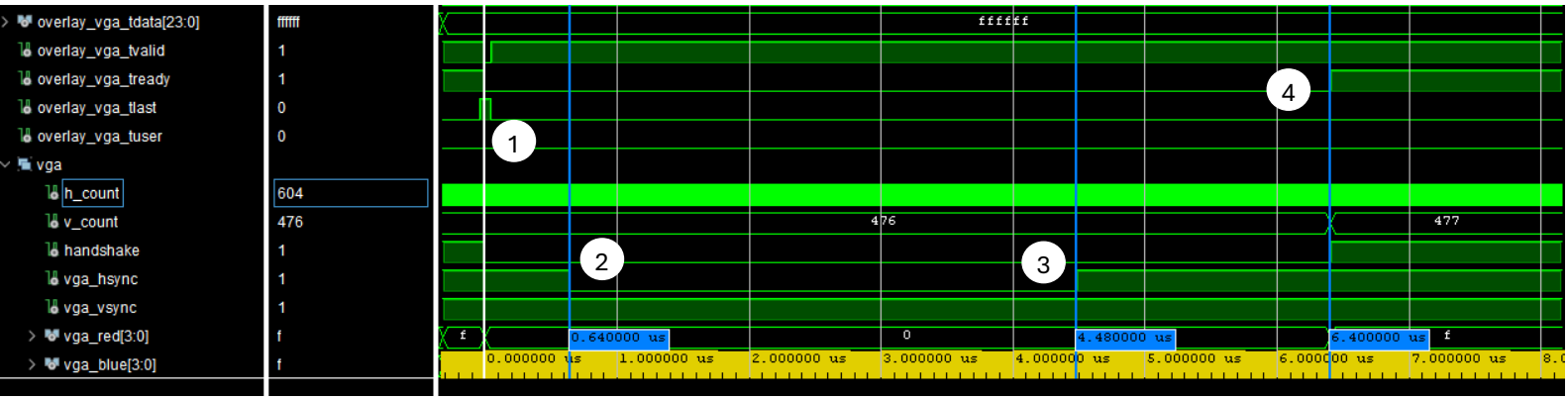
5. Sur le handshake gauss_harris:
La data est transmise au filtre harris 2 coup d'horloge plus tard

6. Sur le handshake filter_overlay et fifo_overlay:
La data est transmise à l'overlay.

7. Sur le handshake overlay_vga:
La data est transmise au convertissuer VGA soit le pixel d'origine seul (fifo) soit remplacé par le résultat du filtrage si l'overlay est activé

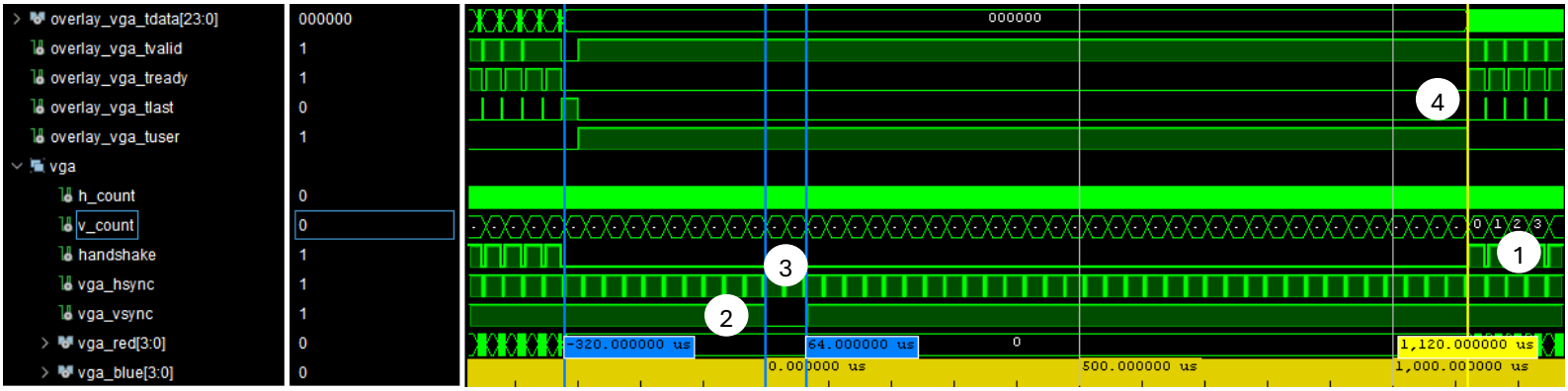
Ligne complète

7.1.2. Validation des timings hsync

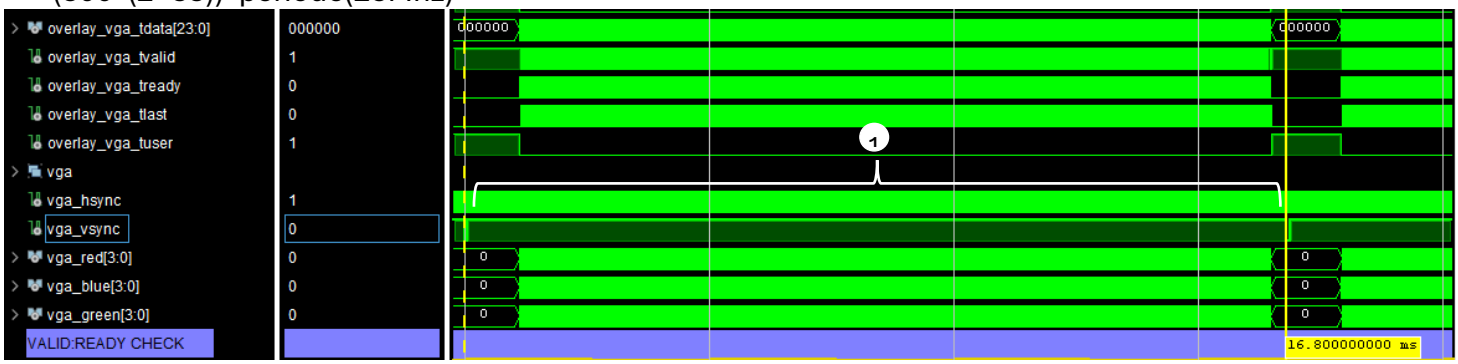


1. Fin de ligne sur l'overlay: overlay_vga_tlast = 1 sur le dernier handshake
2. Le signal hsync passe à 0 0.64us plus tard soit 16* periode(25Mhz)
3. Le signal hsync reste à 0 pendant 96 periode , il repasse à 1 à 4.48us après la fin de ligne
   4.48us = (16+96)* periode(25Mhz)
4. Le protocole VGA dis d'attendre 48 periode supplémentaires avant de lancer la transmission d'un autre ligne
   Le signal tready repasse à 1 à t=6.4us après la fin de ligne pour relancer la transmission d'un autre mot. 6.4us = (16+96+48)* periode(25Mhz)

7.1.3. Validation du timings vsync sur un transmission de frame



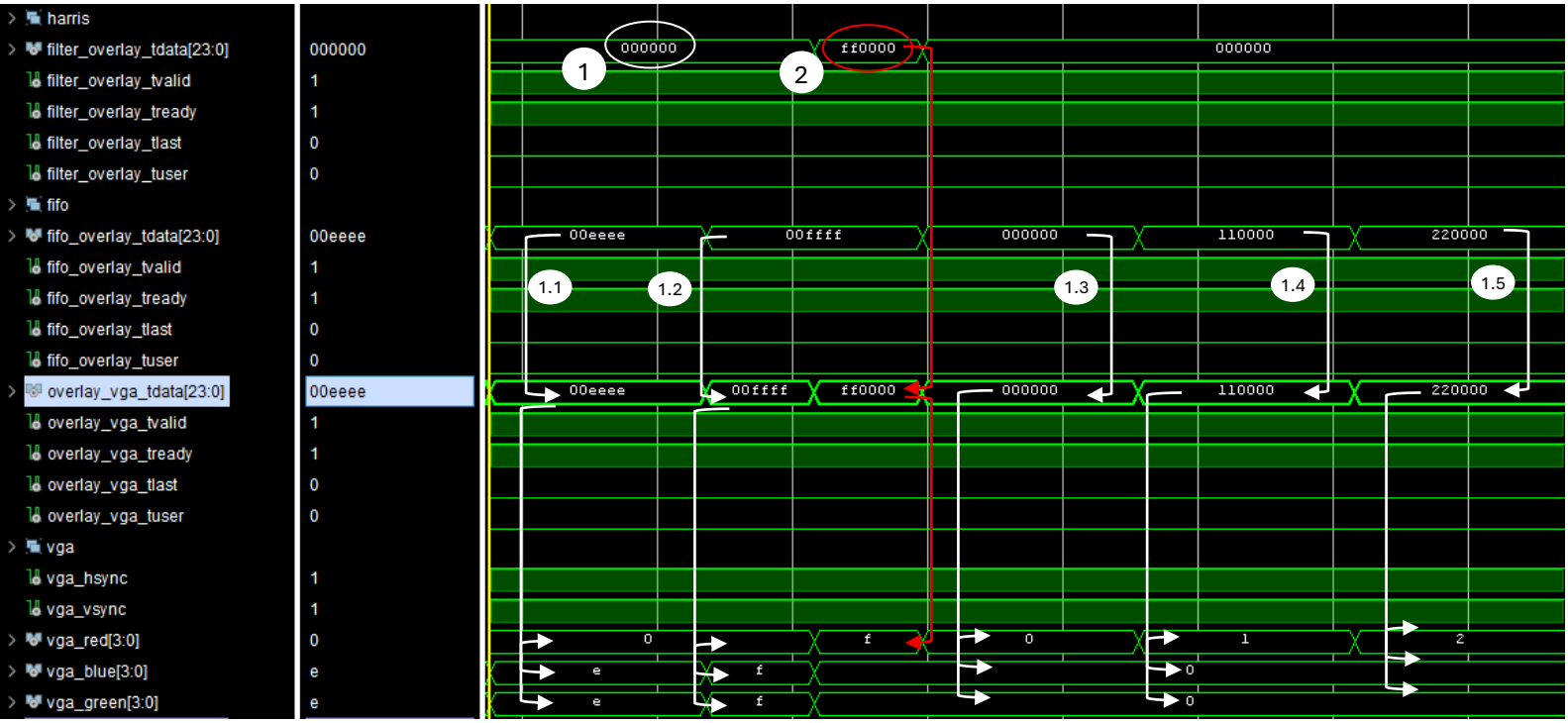
1. Au changement de frame v_count et h_count sont remis à zéro
   Les données commencent à être transférées en commençant par la première ligne (v_count = 0)
2. Le signal vsync attend 10 lignes avant de passer à 0 soit 320us = 800*10*periode(25Mhz)après la fin de frame
3. Le signal vsync reste à 0 pendant 2 lignes, soit 64us = 800*2*periode(25Mhz)
4. La transmission est relancée quand 33 cycles de lignes sont passés à t= 1.120us (800*(2+33))*periode(25Mhz)



1. Selon le protocole VGA , une frame est émise toutes les 16.8ms = 800*525*periode(25Mhz)

Ligne complète

7.1.4. Validation d'un overlay



La ligne est configurée pour autorises l'overlay (add_filter = 1)

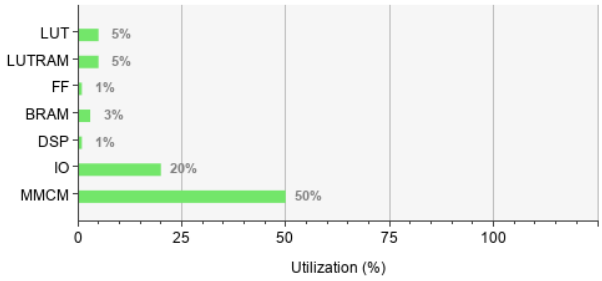
1. Tant que la sortie du filtre harris est à 0, les pixel d'origine sont envoyés sur le VGA (ceux provenant de la fifo) sur le handshake fifo_overlay et filter_overlay
2. La sortie du filtre harris n'est pas nulle , le pixel rouge de contour sortie du filtre harris est envoyés sur le VGA sur le handshake fifo_overlay et filter_overlay

7.2. Test sur carte

7.2.1. Synthèse et implémentation : rapport d'utilisation

Search (Alt+Slash) ame	Slice LUTs (17600)	Slice Registers (35200)	F7 Muxes (8800)	F8 Muxes (4400)	Slice (4400)	LUT as Logic (17600)	LUT as Memory (6000)	Block RAM Tile (60)	DSPs (80)	Bonded IOB (100)	BUFGCTRL (32)	MMCME2_ADV (2)
top_vhdl	897	435	160	64	313	577	320	1.5	1	20	2	1
u_axi4s_to_vga (axi4s_to_vga)	40	21	0	0	13	40	0	0	0	0	0	0
u_broascaster (axis_broadcaster_1to2)	4	2	0	0	1	4	0	0	0	0	0	0
u_clk_wiz_0 (clk_wiz_0)	0	0	0	0	0	0	0	0	0	0	2	1
u_fifo (axis_data_fifo_0)	35	76	0	0	30	35	0	1.5	0	0	0	0
u_filter_gauss (filter_gauss)	359	154	80	32	116	199	160	0	0	0	0	0
u_filter_grey (filter_grey)	17	13	0	0	6	17	0	0	1	0	0	0
u_filter_harris (filter_harris)	297	92	80	32	87	137	160	0	0	0	0	0
u_overlay (axi4s_overlay)	8	0	0	0	4	8	0	0	0	0	0	0
u_pattern_generator (pattern_generator)	138	75	0	0	62	138	0	0	0	0	0	0

Resource	Utilization	Available	Utilization %
LUT	897	17600	5.10
LUTRAM	320	6000	5.33
FF	435	35200	1.24
BRAM	1.50	60	2.50
DSP	1	80	1.25
IO	20	100	20.00
MMCM	1	2	50.00



Ref Name	Used	Functional Category
FDCE	345	Flop & Latch
RAMS64E	320	Distributed Memory
LUT2	240	LUT
LUT6	198	LUT
MUXF7	160	MuxFx
LUT3	121	LUT
LUT4	107	LUT
LUT5	88	LUT
CARRY4	76	CarryLogic
FDRE	67	Flop & Latch
MUXF8	64	MuxFx
LUT1	29	LUT
OBUF	16	IO
FDPE	12	Flop & Latch
FDSE	11	Flop & Latch
IBUF	4	IO
BUFG	2	Clock
RAMB36E1	1	Block Memory
RAMB18E1	1	Block Memory
MMCME2_ADV	1	Clock
DSP48E1	1	Block Arithmetic

Ligne complète

L'utilisation des ressources pour les modules fait maison n'a pas changé .
La fifo et le braodcaster sont des IP xilinxs
L'overlay a été fait maison ,il contient que quelques LUT

7.2.2. Synthèse et implémentation : rapport de timings

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	26,440 ns	Worst Hold Slack (WHS):	0,083 ns	Worst Pulse Width Slack (WPWS):	2,000 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	4092	Total Number of Endpoints:	4092	Total Number of Endpoints:	766

All user specified timing constraints are met.

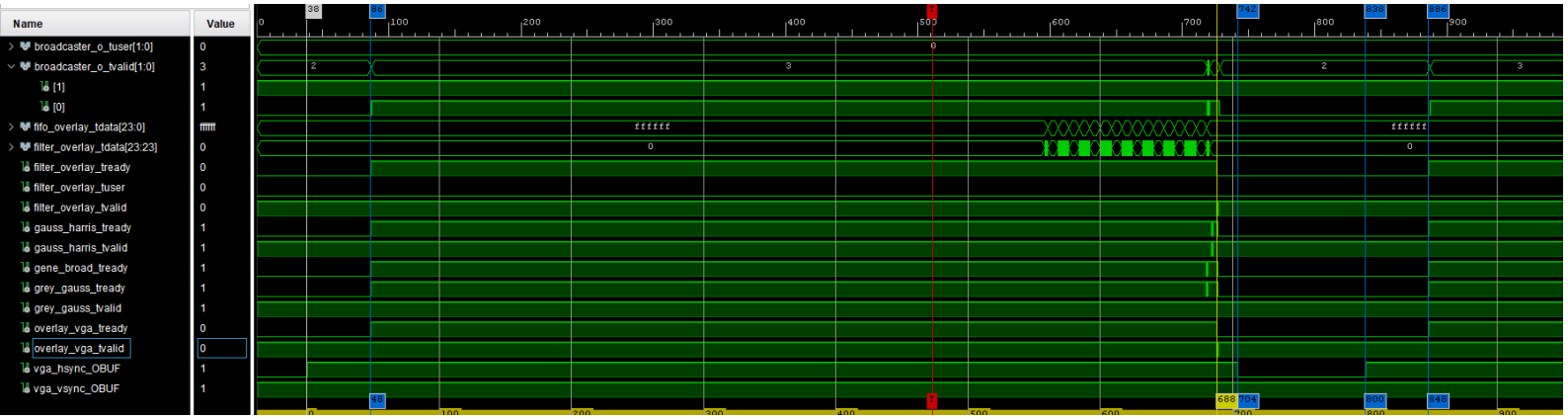
Les contraintes de timing sont respectées avec une horloge à 25Mhz.

Summary

Name	Path 1
Slack	26.440ns
Source	u_pattern_generator/box_x_reg_reg[3]/C (rising edge-triggered cell FDCE clocked by clk_out1_clk_wiz_0 {rise@0.000ns fall@20.000ns period=40.000ns})
Destination	u_filter_grey/multOp/A[8] (rising edge-triggered cell DSP48E1 clocked by clk_out1_clk_wiz_0 {rise@0.000ns fall@20.000ns period=40.000ns})
Path Group	clk_out1_clk_wiz_0
Path Type	Setup (Max at Slow Process Corner)
Requirement	40.000ns (clk_out1_clk_wiz_0 rise@40.000ns - clk_out1_clk_wiz_0 rise@0.000ns)
Data Path Delay	9.458ns (logic 3.942ns (41.681%) route 5.516ns (58.319%))
Logic Levels	12 (CARRY4=6 LUT1=1 LUT2=1 LUT3=1 LUT4=2 LUT6=1)
Clock Path Skew	-0.045ns
Clock Un...rtainty	0.160ns

Toujours le même chemin critique entre les données du pattern generator et la logique de calcul du filtre gris

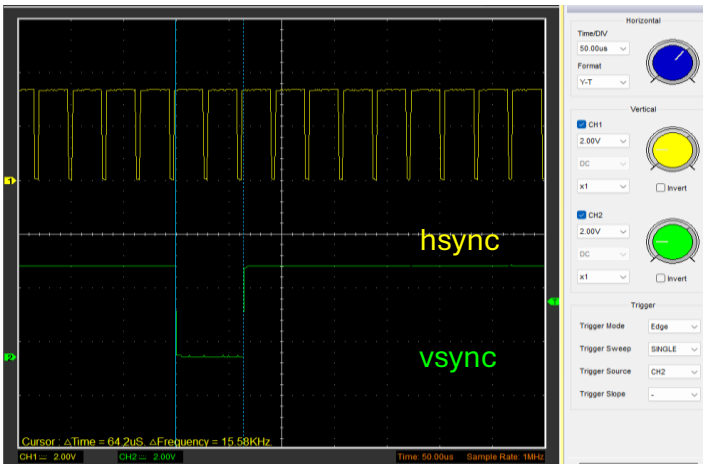
7.2.3. Mesure ILA

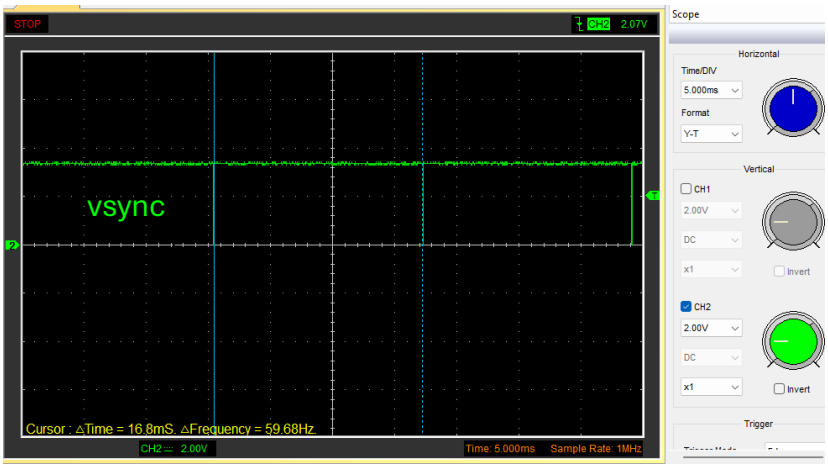


tready reste à 1 sur un ligne pendant 726_86= 640*periode(clk)
un cycle hsync dure 838-38 = 800* periode(clk)
hsync est à 0 pendant 838-742 = 96*periode(clk)
tready passe à 1 886-838 = 48*periode(clk) après que hsync soit repassée à 1
hsync passe à 0 742-726 = 16 cycles après que tready =0

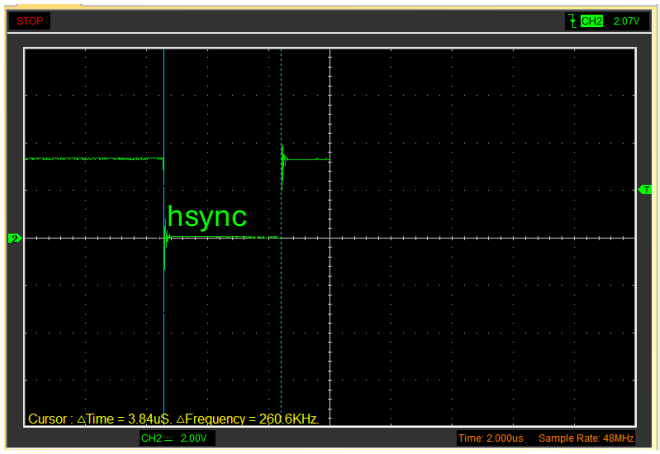
Ligne complète

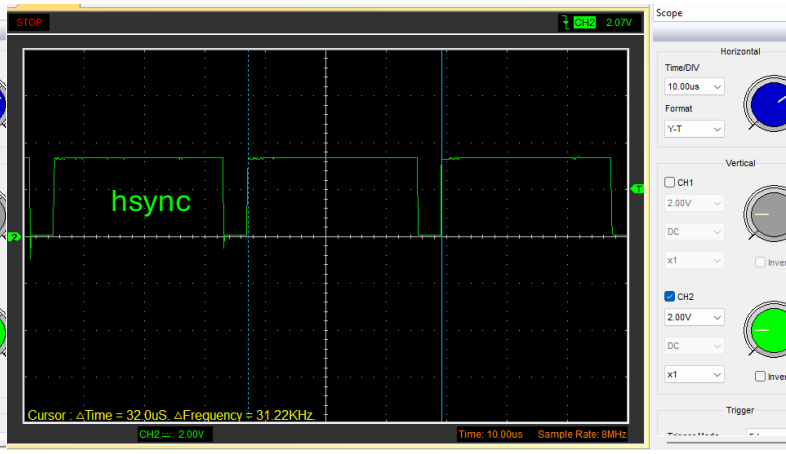
7.2.4. Mesure sur oscilloscope





Le signal vsync passe à 0 toutes les 16.8ms = 800*525* periode(25Mhz)
le signal vsync reste à 0 pendant 2 cycles hsync soit 64us= 800*2*periode(25Mhz)



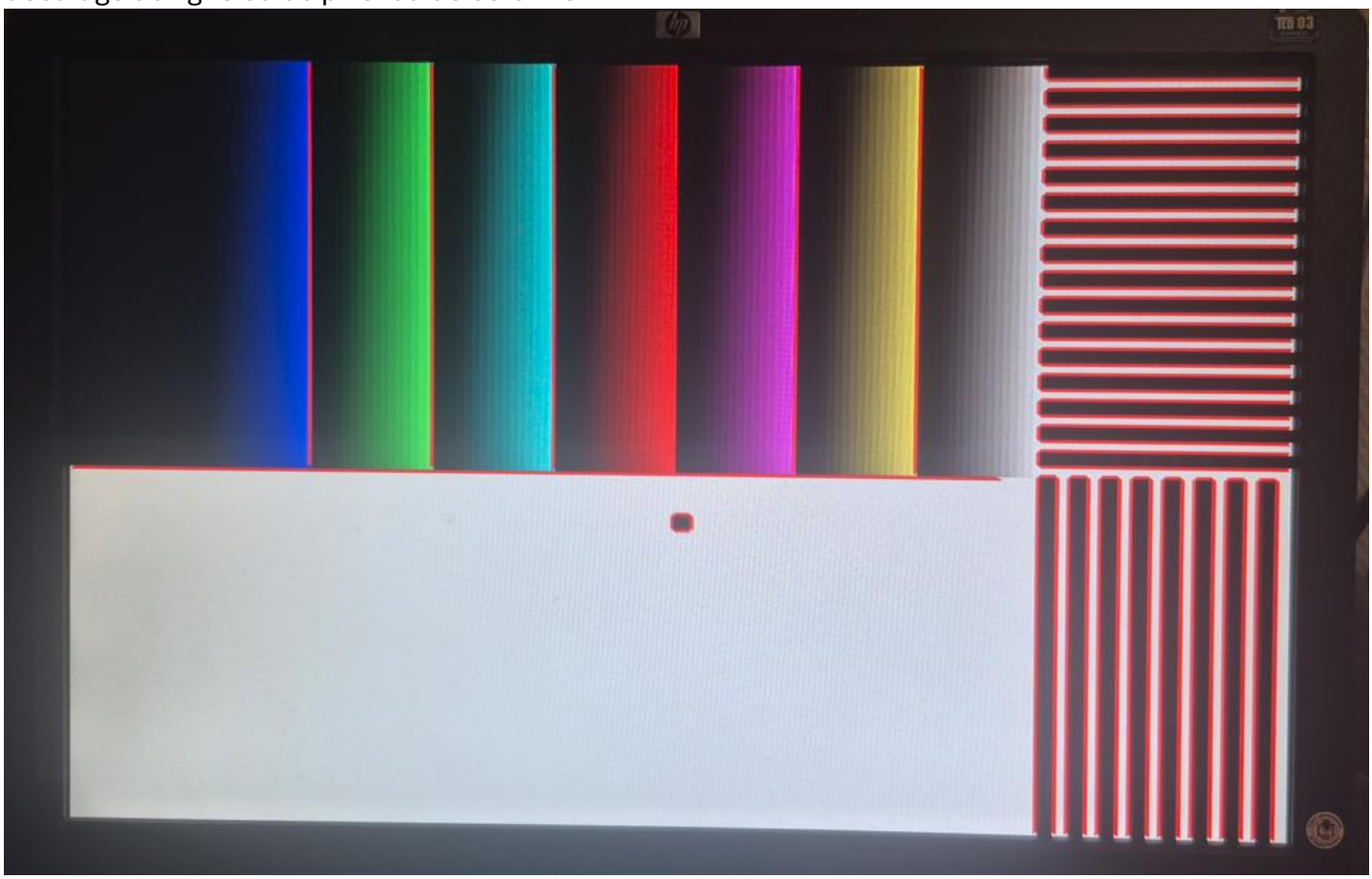


hsync est à 0 pendant 3.83 us ~ 96*periode(25Mhz)
hsync à une période de 32us = 800*periode(clk)

Ligne complète

7.3. Démo

L'image est affichée sur l'écran , on vérifier que celle-ci et stable et ne présenta visuellement de décalage de ligne ou de pixel ou de colonne



Le pattern couleur du départ a bien été filtré, on ne voit plus que les contours du pattern
L'image ne présente pas de décalage de pixel visible.
Le point noir a bien été détecté , son contour se déplace dans le cadre en bas à gauche.

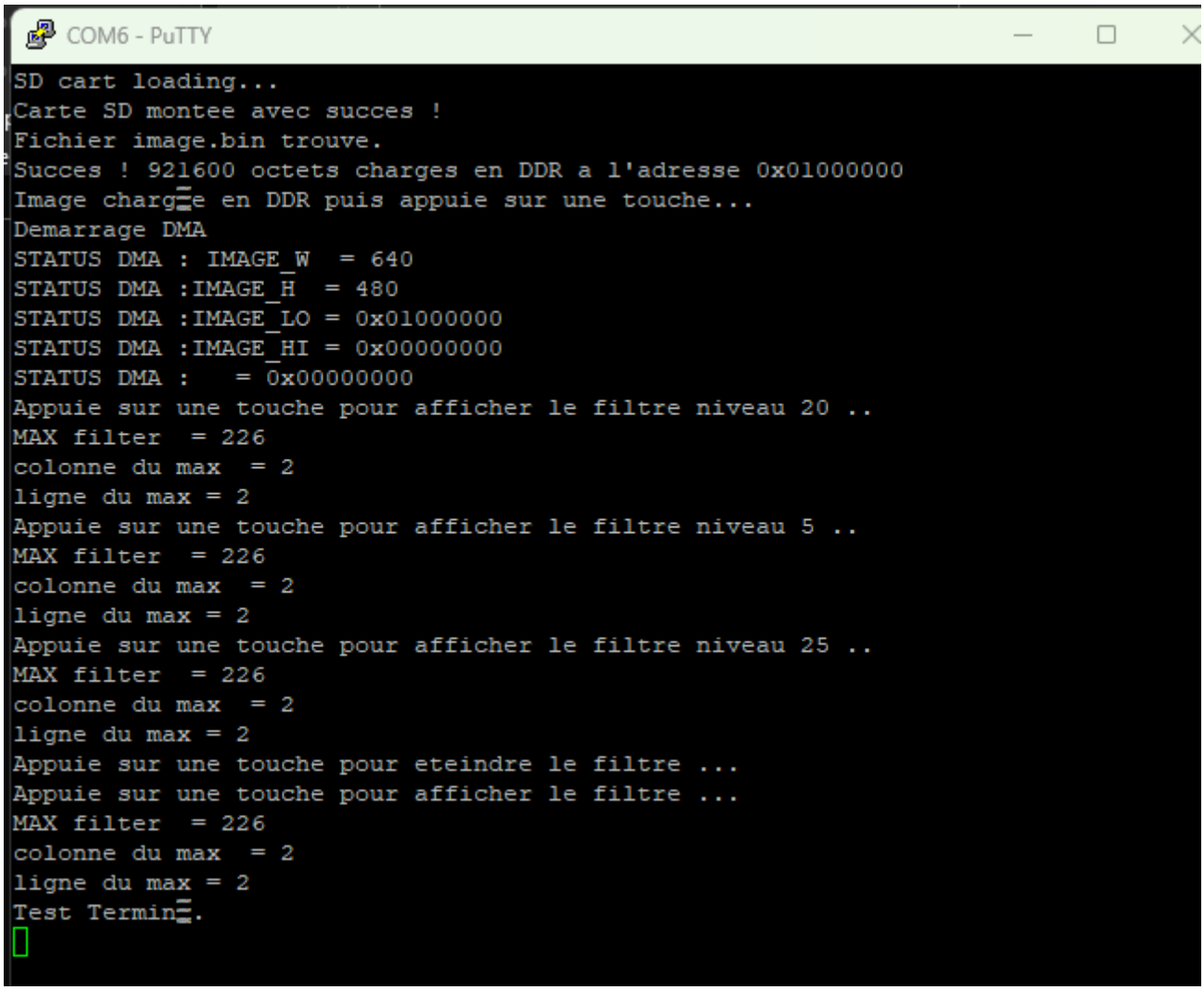
8. Architecture complète

Le fonctionnement du DMA sera vérifiée en même temps que l'architecture complète en affichant l'image stockée dans la DDR

un programme C a été crée pour

- charger l'image de la carte SD dans la DDR
    - on vérifier si la mémoire est bien écrite
- allumer le DMA
    - on vérifier l'affichage de l'image sur l'écran
- afficher le filtre
    - on vérifier l'affichage de l'image sur l'écran
- changer la valeur le filtre
    - on vérifier l'affichage de l'image sur l'écran

Les informations sont affichée sur l'uart



La communication avec l'uart nous permet de dialoguer avec le circuit en choisissant le niveau de filtrage souhaité, l'affichage ou non du filtrage.

Architecture complète

L'image est affichée sans filtre au lancement du programme à partir de l'image chargeé dasn la carte SD

L'affichage de l'overlay sur l'image est fonctionnel

Architecture complète

Si on augmente le niveau de l'overlay, les contours seront moins dessinés

Affichage de pattern

```
Affichage du pattern ..
Appuie sur une touche pour afficher le filtre niveau 20 ..
MAX filter  = 572
colonne du max  = 1
ligne du max = 10
Appuie sur une touche pour afficher le filtre niveau 5 ..
MAX filter  = 572
colonne du max  = 1
ligne du max = 10
Appuie sur une touche pour afficher le filtre niveau 25 ..
MAX filter  = 572
colonne du max  = 1
ligne du max = 10
Appuie sur une touche pour eteindre le filtre ...
Appuie sur une touche pour afficher le filtre ...
```

L'affichage du pattern et de son filtrage fonctionne aussi correctemen
Il manquera la validation du switch entre les deux sources d'image à la volée pour le code C .
Actuellement l'intégration de l'ip axi4s switch dans le système n'est pas fine de valider

9. Statut de validation

	Développé	Validé	Fonctionnel
2. FPGA image processing			
2.1. [PL-IP-000] Source Vidéo mémoire [1]			
2.2. [PL-IP-001] Source Vidéo TPG [1]			
2.3. [PL-IP-002] Sélection de source vidéo init [1]			
2.4. [PL-IP-003] Sélection de source vidéo à chaud [2]			
2.5. [PL-IP-004] Architecture filtre à convolution [1]			
2.6. [PL-IP-005] Standards de transferts [1]			
2.7. [PL-IP-006] Filtre gaussien [1]			
2.8. [PL-IP-007] Filtre Harris [1]			
2.9. [PL-IP-008] Filtre Harris binarisation [1]			
2.10. [PL-IP-009] Filtre Harris seuillage paramétrable [1]			
2.11. [PL-IP-010] Filtre Harris retour de données [2]			
2.12. [PL-DISP-001] Overlay [1]			
2.13. [PL-DISP-002] Affichage écran VGA [1]			
2.14. [PL-DISP-003] Mode d’affichage [2]			
3. SoC			
3.1. [PS-001] Mise en mémoire manuelle pattern [1]			
3.2. [PS-002] Mise en mémoire automatique pattern [2]			
3.3. [PS-003] Retour de données UART [2]			

10. Conclusion

Les éléments de l'architecture ont pu être individuellement puis intégré dans le système.
La validation a été faite en simulation puis en test sur carte.
Tous les éléments de la chaine de filtrage sont fonctionnels. Leur configuration par sofware est fonctionnelle .
Il reste le switch entre les deux fournisseurs d'image en entrée du système qui n'est pas fini de développer et de valider.