Rapport TP 04 partie 2

Samory DIABY

Table des matières

1 Objectifs

L'objectif de cette partie est de réaliser un design permettant de faire clignoter une LED RGB avec une séquence de couleurs entrées à l'aide des boutons. Dans cette partie, vous utiliserez le module LED_driver de la partie 1 et vous ajouterez l'utilisation d'un composant mémoire : la FIFO.

2 Notes

Pour ce tp, j'ai assumé que les compteurs étaient codés sur 28 bits.

3 Réalisations

3.1 Modification du module led_driver de la partie 1

Dans un premier temps il faut modifier le module **led_driver** pour lui ajouter la logique liée à la nouvelle sortie **end_cycle**. Voici le schéma RTL associé à ces changements :

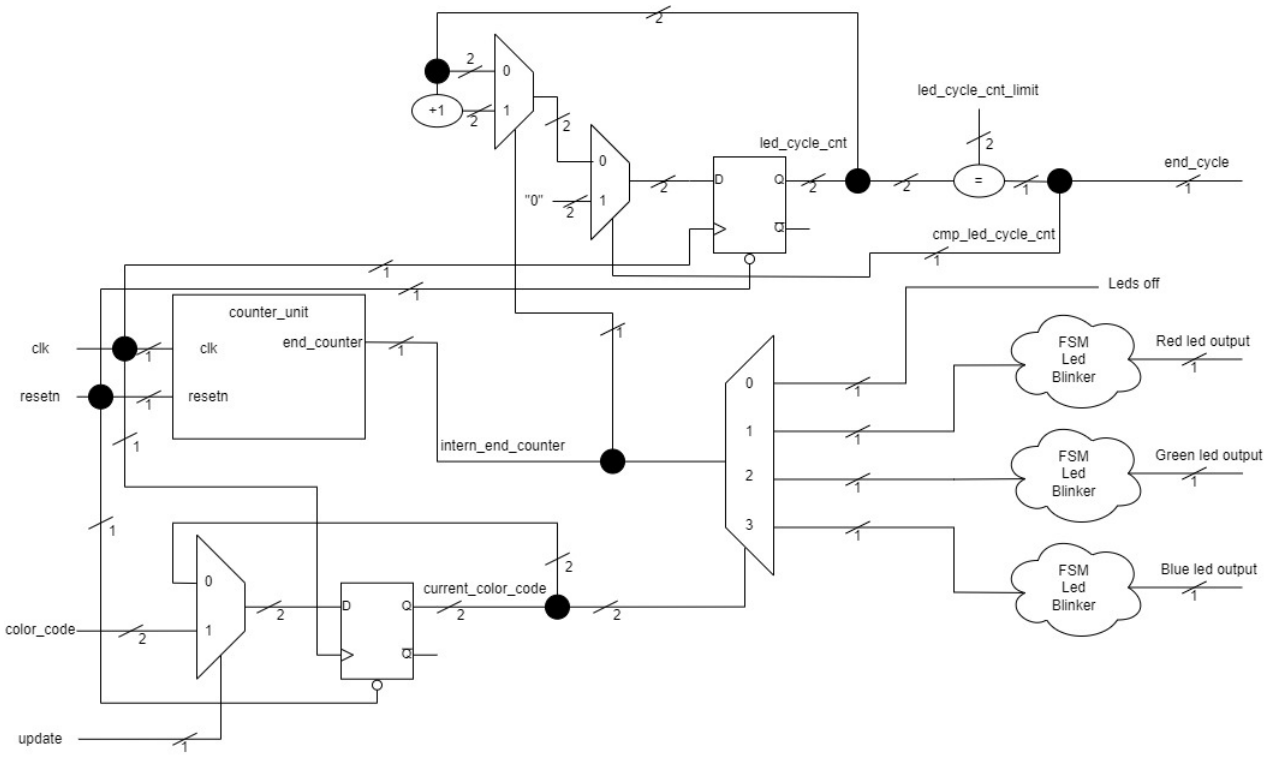

Figure 1 – Schéma RTL du nouveau module led_driver

Sur le schéma ci-dessus on peut voir l'ajout d'un nouveau compteur qui va compter 2 fronts montants du signal **end_counter** avant de repasser à "0".

3.2 Ajout de la FIFO pour contrôler le code couleur

On souhaite maintenant ajouter une FIFO pour contrôler le code couleur passé en entrée du module led_driver. On modifie donc le schéma RTL pour ajouter notre FIFO :

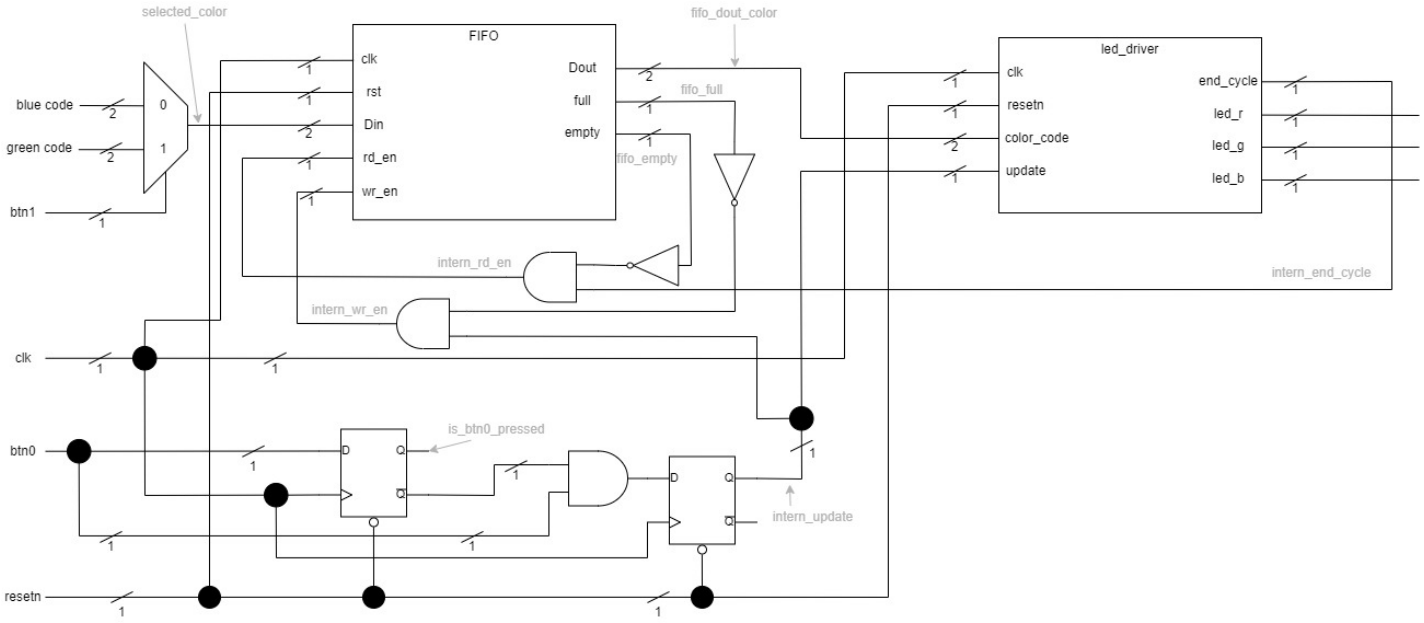

Figure 2 – Nouveau schéma RTL avec FIFO

L'entrée **color_code** du module led_driver prends maintenant la sortie **Dout** de la FIFO, et la FIFO prends en entrée **Din** le signal **selecetd_color** qui sera écrite dans cette même FIFO lorsque le signal **intern_wr_en** sera à "1" (qui lui même sera à "1" quand la FIFO n'est pas **full** et que le signal **intern_update** est à "1").
La lecture s'effectue de manière similaire à l'écriture : on lit une valeur dans la FIFO quand elle n'est pas vide (le sginal **empty** est à "0") et que le signal **end_cycle** est à "1".

La logique pour les signaux **intern_update** et **selected_color** est la même que pour la première partie du TP04.

3.2.1 Configuration de la FIFO

Voici la configuration utilisée dans ce TP pour notre FIFO :

— **Basic** :
  Interface type : Native
  FIFO Implementation options : Common Clock (BRAM)
— **Native Ports** :
  Read mode : Standard FIFO
  Data Port Params :
    write width : 2
    write depth : 512
    read width : 2
    read depth : 512
  Initialization :
    Reset pin : true
    reset type : synchronous
    Full flags reset value : 0
    Dout reset value : 0

3.2.2 Fonctionnement du circuit

Pour décrire le fonctionnement du circuit nous allons nous servir du chronogramme ci-dessous :

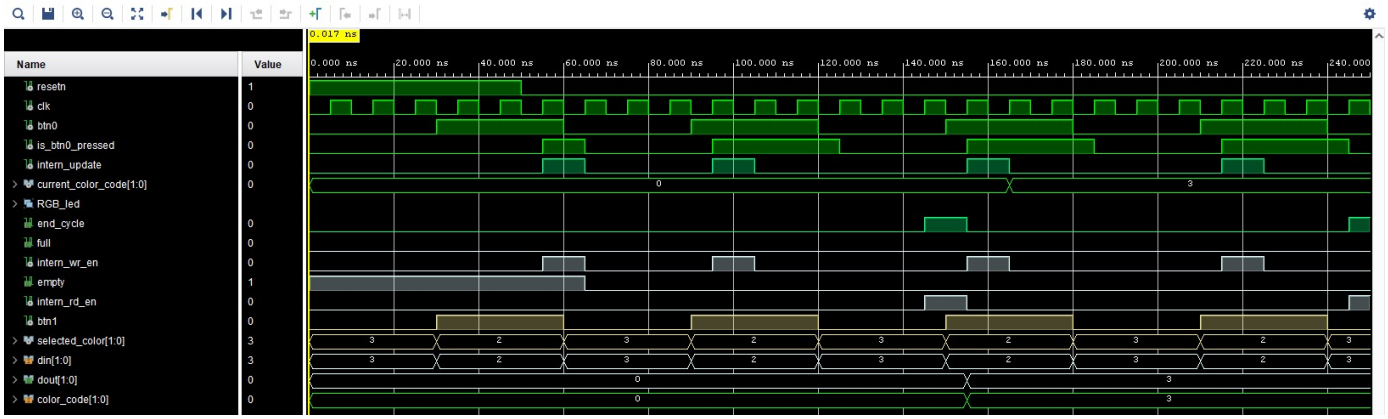

Figure 3 – Chronogramme lié au schéma section 3.2 (testbench Q4)

En commençant à t = 0s, on peut voir que le signal **resetn** est à "1" pendant 50ns, ce qui force le système à tout initialiser à "0".

A partir de t = 55ns, le signal **intern_update** passe à "1" pendant un appui sur le bouton **btn0**. Lorsque le signal **intern_update** passe à 1, le signal **intern_wr_en** passe également à 1 car la FIFO est vide, ce qui va permettre d'écrire la valeur actuelle du signal **selected_color**, qui vaut "3" au moment de l'écriture (front montant à t = 65ns), dans la

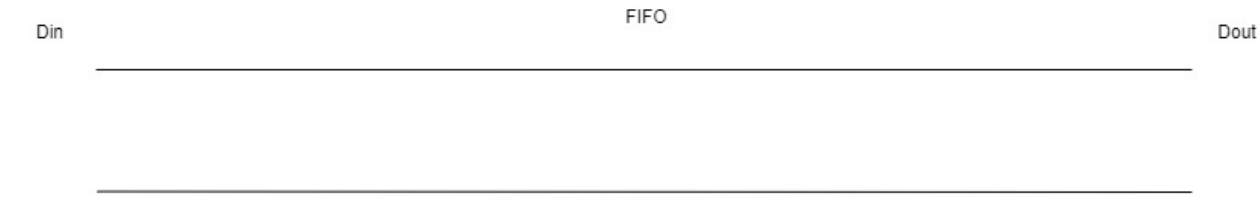

Figure 4 – FIFO - vide

FIFO via l'entrée **Din**. On a alors la valeur "3" dans la FIFO.

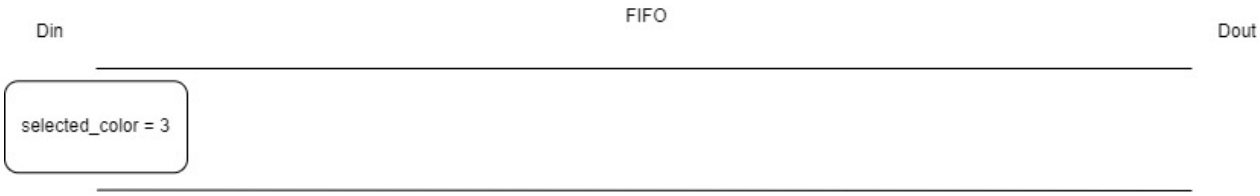

Figure 5 – FIFO - première insertion de selected_color

On voit que le signal **intern_wr_en** repasse à "1" à t = 95ns, ce qui permet d'écrire la valeur "2" de **selected_color** dans la FIFO.

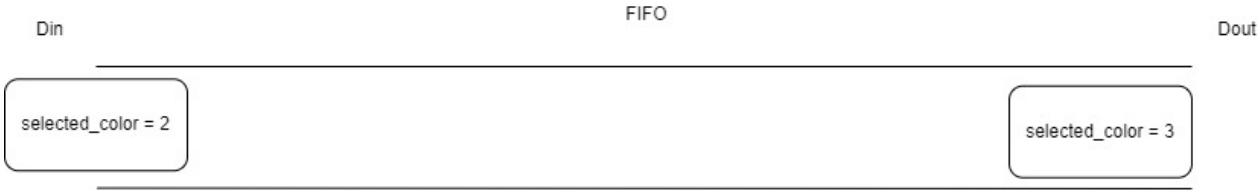

Figure 6 – FIFO - deuxième insertion de selected_color

On a donc 2 valeurs dans notre FIFO : "3" et "2".

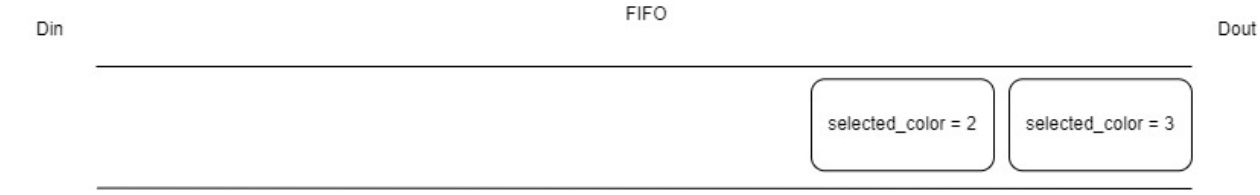

Figure 7 – FIFO - 2 valeurs stockées

On remarque maintenant qu'à t = 145ns, le signal **end_cycle** passe à "1", ce qui va entraîner une lecture de la FIFO au prochain front montant, ce que l'on remarque à t = 155ns avec le changement de valeur du signal **Dout** qui passe de "0" à "3" (première valeur inscrite dans la FIFO).

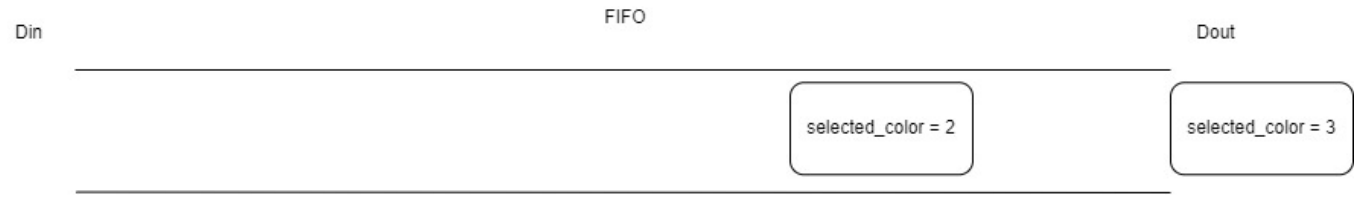

Figure 8 – FIFO - lecture

La valeur lue, ici "3", va être stocké dans le registre interne de notre "standard FIFO" et va faire en sorte que cette dernière valeur lue reste jusqu'à une prochaine lecture valide dans la FIFO.

3.3 Analyse du rapport de synthèse

A partir du rapport de synthèse, on peut observer dans la section "RTL components statistics" que l'on a 2 registres 2 bits qui correspondent aux registres du compteurs **end_cycles** et à celui de la sauvegarde de la couleur courante (**current_color_code**) du module **led_driver**. Les 3 registres 1 bit sont : **current_state_reg** qui stocke l'état de la FSM, **is_btn0_pressed** et **intern_update** qui gèrent la détection de front montant du bouton 0.
Le composant **adder** ne fait pas parti du schéma RTL et doit faire parti de la FIFO.

```
---------------------------------------------------------------------------------
Start RTL Component Statistics
---------------------------------------------------------------------------------
Detailed RTL Component Info :
+---Adders :
      2 Input    2 Bit       Adders := 1
+---Registers :
                   2 Bit    Registers := 2
                   1 Bit    Registers := 3
+---Muxes :
      2 Input    2 Bit        Muxes := 2
      2 Input    1 Bit        Muxes := 2
---------------------------------------------------------------------------------
Finished RTL Component Statistics
---------------------------------------------------------------------------------
```

Figure 9 – Rapport de synthèse - composants RTL

En regardant la section "Cell usage", on peut retrouver nos 35 registres (FDCE) qui sont répartis de cette manière :

- 28 FDCE pour le compteur de **counter_unit**
- 4 FDCE pour les 2 registres 2 bits décrit précédemment
- 3 FDCE pour les 3 registres 1 bit décrit précédemment

3.4 Analyse du rapport de timing

A partir du rapport de timing, on peut observer que l'on a aucune violation de setup ou ni de hold (via la section **Design timing summary**).

Report Cell Usage:

	Cell	Count
1	fifo_generator	1
2	BUFG	1
3	CARRY4	7
4	LUT1	1
5	LUT2	3
6	LUT3	6
7	LUT4	4
8	LUT5	29
9	LUT6	4
10	FDCE	35
11	IBUF	4
12	OBUF	3

Figure 10 – Rapport de synthèse - Cell uasage

Design Timing Summary

WNS(ns)	TNS(ns)	TNS Failing Endpoints	TNS Total Endpoints	WHS(ns)	THS(ns)	THS Failing Endpoints	THS Total Endpoints
5.152	0.000	0	134	0.154	0.000	0	134

Figure 11 – Rapport de timing - Design Timing summary

On retrouve également le chemin critique donné par vivado avec le **Max Path** de la section **Timing details**. Le chemin critique de notre design est le chemin qui qui va jusqu'au compteur interne de la FIFO (qui compte le nombre d'éléments présent dans la FIFO).

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             5.152ns  (required time - arrival time)
  Source:                 FIFO_ISNT/U0/inst_fifo_gen/gconvfifo.rf/grf.rf/gntv_or_sync_fifo.gl0.rd/rpntr/gc0.count_d1_reg[2]/C
                            (rising edge-triggered cell FDRE clocked by sys_clk_pin  {rise@0.000ns fall@5.000ns period=10.000ns})
  Destination:            FIFO_ISNT/U0/inst_fifo_gen/gconvfifo.rf/grf.rf/gntv_or_sync_fifo.gl0.wr/gwss.wsts/ram_full_fb_i_reg/D
                            (rising edge-triggered cell FDRE clocked by sys_clk_pin  {rise@0.000ns fall@5.000ns period=10.000ns})
```

Figure 12 – Rapport de timing - Chemin critique

3.5 Résultats de l'ILA

Pour vérifier le fonctionnement du design, on utilise l'ILA. Le chronogramme ci-dessous montre le résultat de l'ILA lors du tout premier appui sur le bouton 0 (juste après l'envoi du code sur la carte). On peut voir que le signal **intern_wr_en** passe à "1".

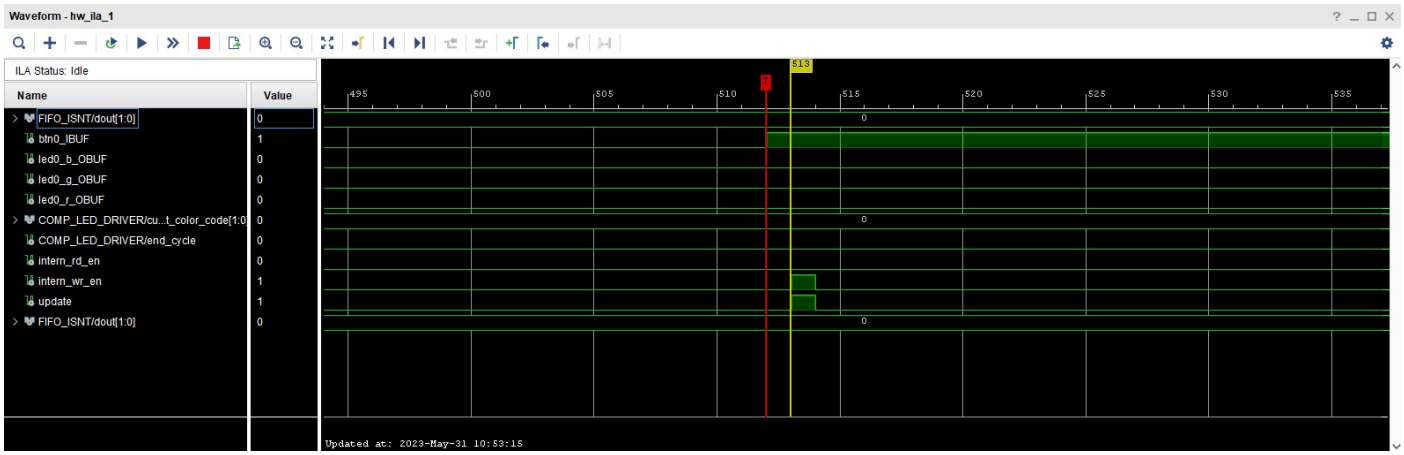

Figure 13 – ILA - 1er appui sur le bouton 0

On observe ensuite une lecture de la FIFO lorsque le signal **end_cycle** passe à "1" ce qui change la sortie de la FIFO de "0" à la valeur précédemment écrite, ici "3".

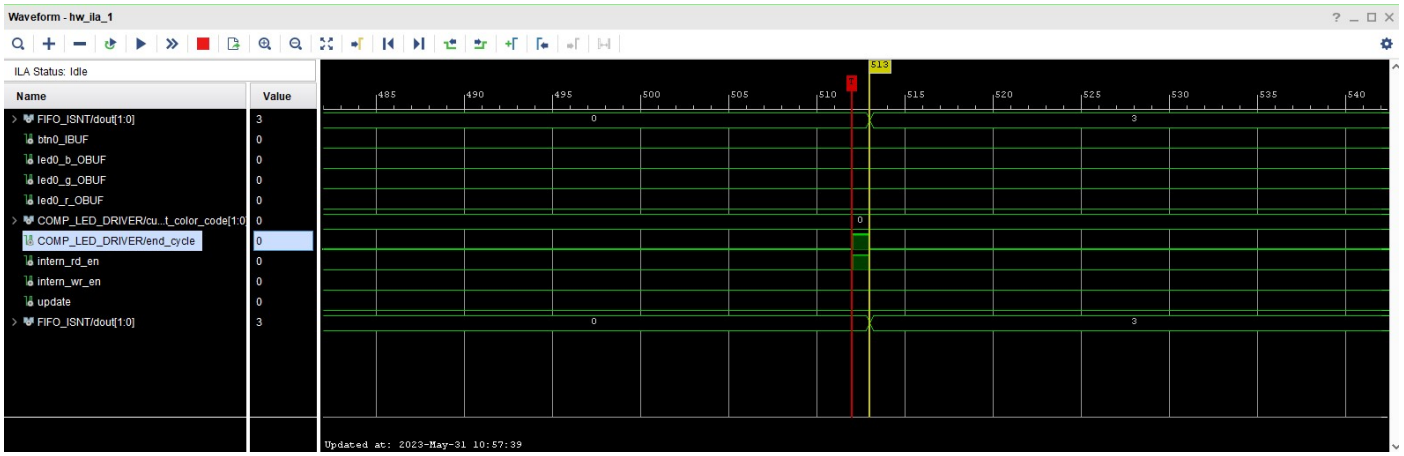

Figure 14 – ILA - lecture de la FIFO

Ensuite, on appui une seconde fois sur le bouton 0, ce qui provoque une mise à jour de la couleur dans le composant **led_driver** (que l'on remarque avec le changement du signal **current_color_code**). L'appui va également provoquer uen nouvelle écriture dans la FIFO avec le passage de **intern_wr_en** à "1".

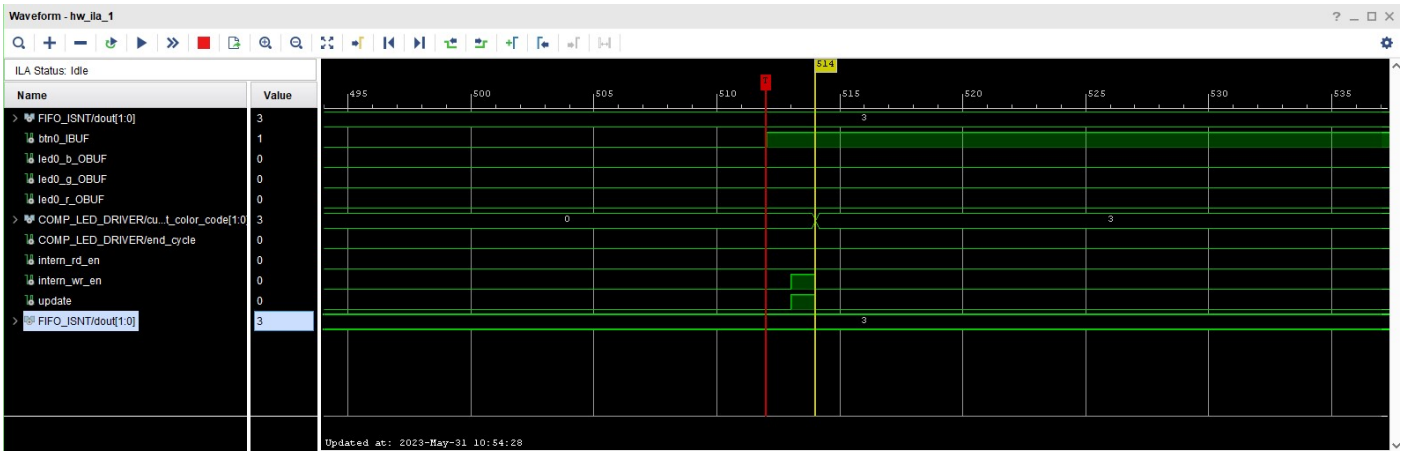

Figure 15 – ILA - 2ème appui sur le bouton 0