Houssein MARIAM

TP03 – Machine à états (FSM)

**Objectif**

Le présent rapport porte sur la réalisation d'une architecture permettant de contrôler le clignotement d’une LED RGB en utilisant les couleurs rouge, vert et bleu. Ce TP a pour objectif de mettre en pratique l'utilisation de machines à états pour piloter les LEDs, tout en explorant l'utilisation de paramètres génériques et de modules. L'implémentation de cette architecture nous permettra d'acquérir des connaissances pratiques en conception matérielle, en utilisant des concepts avancés tels que la modularité et la flexibilité offertes par les paramètres génériques.

**1.Dans un fichier *.vhd*, créez un module *Counter\_unit* à partir du compteur du TP1. Le module prendra en entrée un signal d’horloge et de resetn, et donnera en sortie le signal *end\_counter*. Utilisez un paramètre *generic()* pour définir le nombre de coup d’horloge à compter.**

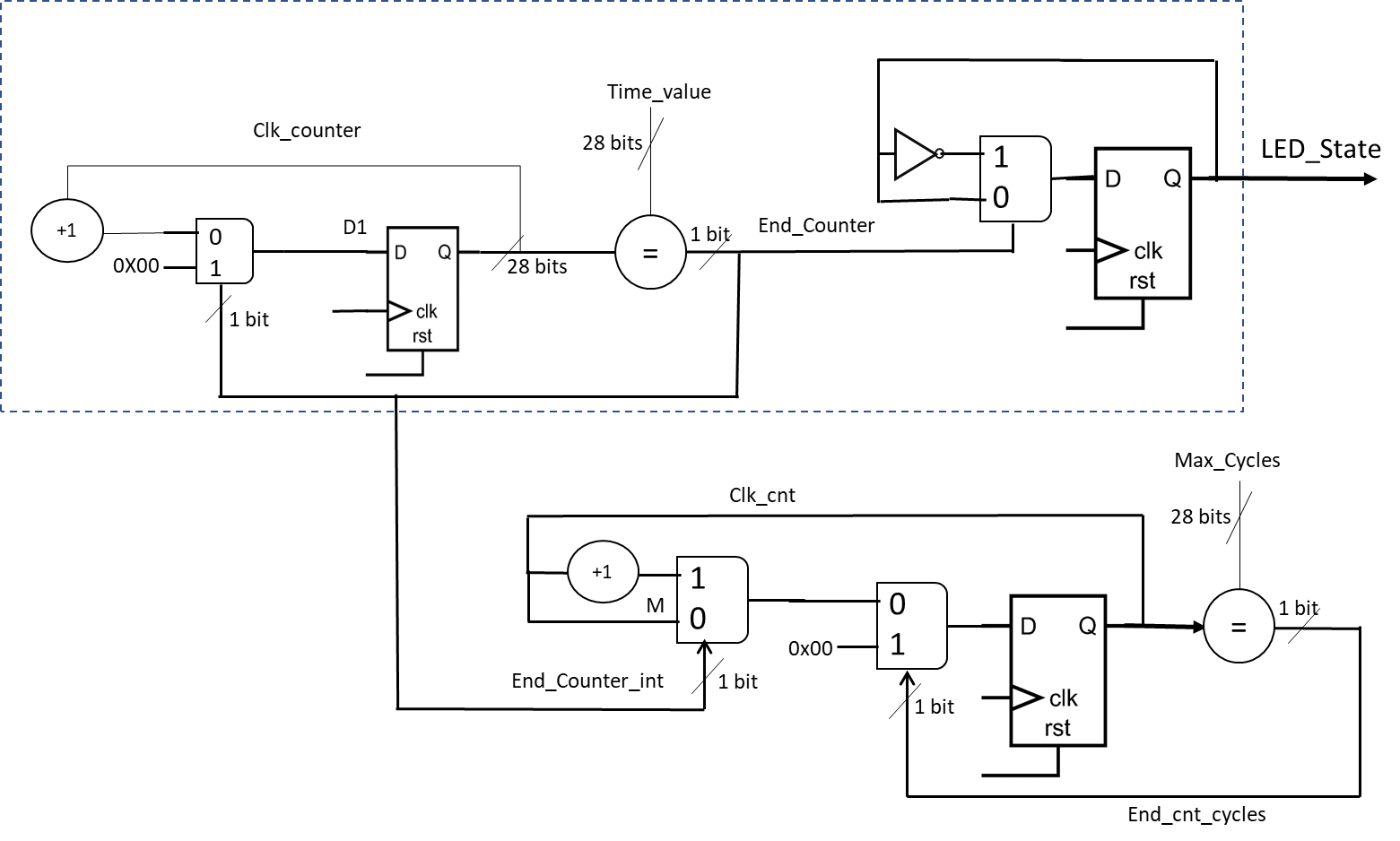
Le code VHDL correspondant est :

Une image contenant texte, capture d’écran

Description générée automatiquement

**2. En schéma RTL, créez un compteur du signal *end\_counter*. Ce compteur doit permettre de déterminer le nombre de cycles allumé/éteint qui ont été effectués par la LED. Le compteur doit pouvoir être remis à 0, maintenir sa valeur actuelle ou s’incrémenter.**

Le compteur du signal end\_counter est un élément essentiel de notre schéma RTL. Il permet de suivre le nombre de cycles allumé/éteint effectués par la LED. Ce compteur peut être réinitialisé à zéro, maintenir sa valeur actuelle ou s'incrémenter. Grâce à ce compteur, nous sommes en mesure de mesurer et contrôler précisément le comportement de la LED dans notre système.

****

**3. Ecrivez un code VHDL décrivant ce compteur de cycle, vous utiliserez le module *Counter\_unit*.**

- Le module Counter\_unit :

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

- Le compteur de cycle :

Une image contenant texte, capture d’écran, document

Description générée automatiquement

**4. Tester votre architecture avec un testbench.**

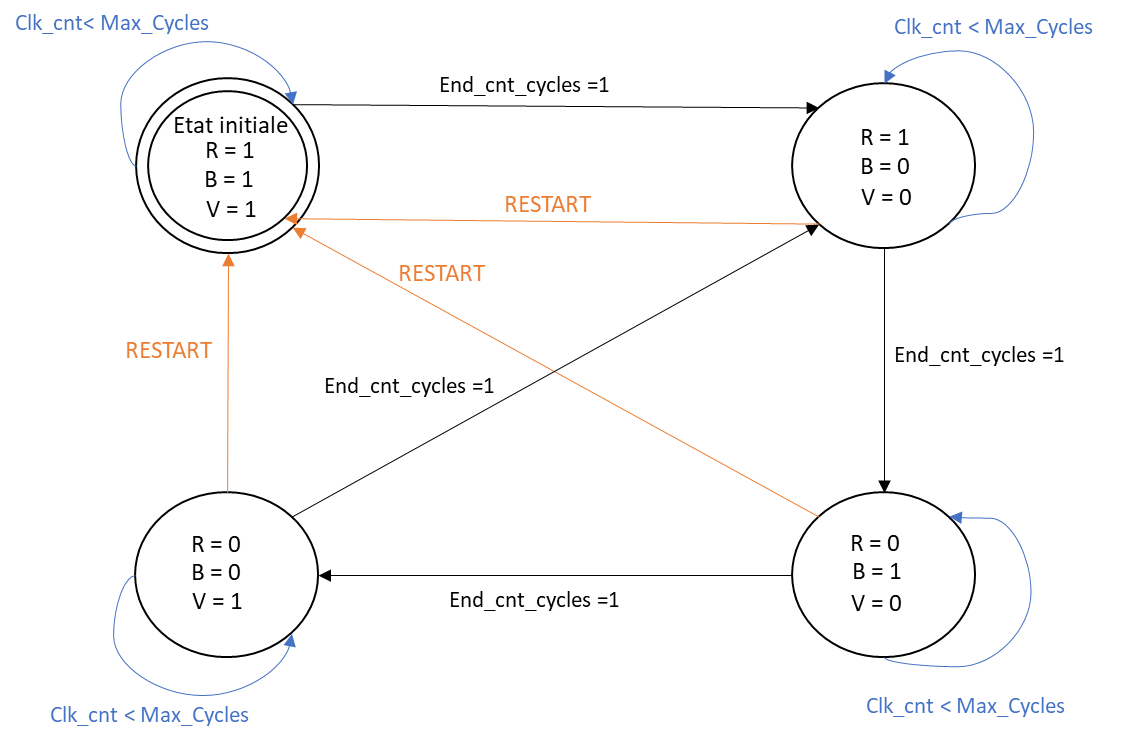
Selon les résultats de notre testbench, le signal "Clk\_cnt" joue un rôle essentiel dans la détermination du nombre de cycles allumé/éteint effectués par la LED. Le compteur s'incrémente lorsque le signal "end\_counter\_int" du module "counter\_unit" atteint sa valeur maximale prédéfinie. Ce signal est représenté par le chronogramme en bleu, qui prend la valeur 1 lorsque le "time\_value" atteint sa valeur maximale fixée à 6. Le compteur maintient sa valeur et s'incrémente lorsque "end\_counter\_int" est égal à 1, jusqu'à atteindre 4 cycles (max\_cycles = 4), puis il est remis à zéro pour recommencer le compte.

Une image contenant capture d’écran, Logiciel multimédia, logiciel, Logiciel de graphisme

Description générée automatiquement

**5. Créez en RTL une machine à états (FSM) permettant de faire clignoter une LED RGB en rouge puis bleu et enfin en vert avant de recommencer le cycle (rouge, bleu, vert, …). Dans chaque état la LED devra clignoter 3 fois. De plus, si le bouton restart est appuyé, on retourne dans l’état initial quel que soit l’état dans lequel on se situe. L’état initial est l’état dans lequel on se situe au démarrage, on passe à l’état rouge après 3 clignotements de la LED en blanc (rouge, vert et bleu actifs en même temps).**

Pour répondre à cette question, nous avons créé une machine à états (FSM) en RTL qui permet de faire clignoter une LED RGB en suivant le cycle suivant : rouge, bleu, vert, puis recommencer. Chaque état de la machine correspond à une couleur spécifique et la LED clignote 3 fois dans chaque état. De plus, si le bouton "restart" est appuyé, la machine retourne à l'état initial, quel que soit l'état dans lequel elle se trouve. L'état initial est défini comme l'état dans lequel la machine se trouve au démarrage. Pour passer de l'état initial à l'état rouge, la LED clignote en blanc avec les composantes rouge, vert et bleu actives en même temps pendant 3 cycles.

****

**6. Listez les signaux d’entrée, de sortie et les signaux internes de votre architecture.**

|  |  |  |
| --- | --- | --- |
| Signaux d’entrée | Signaux de sortie | Signaux internes |
| CLK | OUT\_R | LED\_State |
| Resetn | OUT\_B | End\_Counter\_int |
| Restart | OUT\_V | End\_cnt\_cycles |
|  |  | Clk\_cnt |
|  |  | Max\_Cycles |
|  |  | current\_state |
|  |  | next\_state |

**7. Ajoutez à votre code VHDL les éléments que vous venez de créer.**

La partie FSM :

Une image contenant texte, capture d’écran

Description générée automatiquement

Une image contenant texte, capture d’écran, nombre

Description générée automatiquement

**8. Ecrivez un testbench pour tester votre architecture. Vérifiez à la simulation que vous obtenez le résultat attendu.**

Le résultat de notre testbench est présenté dans la capture d'écran ci-dessous. Au démarrage du système, la LED clignote deux fois en blanc, puis passe à l'état rouge après deux clignotements, puis à l'état bleu après deux clignotements, et enfin à l'état vert. Nous pouvons observer l'état actuel de la machine ainsi que l'état suivant dans lequel nous nous trouvons. De plus, le nombre de cycles allumé/éteint est également indiqué dans le chronogramme en couleur orange. Ainsi, nous avons validé notre architecture avec succès.

Une image contenant capture d’écran, texte, affichage, logiciel

Description générée automatiquement

**9. Exécutez la synthèse et relevez les ressources utilisées (y compris la FSM). Sur la schématique, identifiez où se situe votre compteur de cycle.**

L'analyse des résultats nous permet de tirer plusieurs informations importantes. Tout d'abord, nous constatons qu'un automate à états finis (FSM) a été inféré pour le registre d'état 'current\_state' dans le module 'tp\_fsm'. Cela confirme que notre conception a été reconnue comme un FSM par l'outil de synthèse.

En examinant la table d'encodage des états, nous pouvons voir les nouvelles encodages associées à chaque état ainsi que les encodages précédentes. L'état 'idle' est représenté par l'encodage '00', l'état 'rouge' par '01', l'état 'bleu' par '10' et l'état 'vert' par '11'. Cette table nous aide à comprendre comment les états sont encodés et comment ils sont associés entre eux.

Une image contenant texte, reçu, ligne, capture d’écran

Description générée automatiquement

En ce qui concerne l'utilisation des cellules de la FPGA, le rapport indique le nombre de chaque type de cellule utilisée dans notre conception. Nous pouvons observer différentes cellules telles que des BUFG (buffers d'horloge), des LUTs (tables de vérité), des registres FDCE et des buffers d'entrée/sortie IBUF/OBUF. Cette information est utile pour évaluer l'efficacité de l'utilisation des ressources de la FPGA par notre conception.

Une image contenant texte, capture d’écran, nombre, reçu

Description générée automatiquement

En conclusion, l'analyse des résultats nous donne un aperçu approfondi de notre conception. Nous avons réussi à inférer un FSM, nous avons identifié les encodages des différents états, et nous avons évalué l'utilisation des cellules dans notre conception. Ces informations nous aident à vérifier la validité de notre conception et à optimiser notre utilisation des ressources matérielles.

La schématique ci-dessous permet d'identifier facilement notre module cunter\_unit, le FSM (FSM\_sequential\_current\_state), le compteur de cycles (Clk\_cnt\_reg), ainsi que les entrées et sorties de notre architecture.

Une image contenant diagramme, Plan, Dessin technique, schématique

Description générée automatiquement

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

**10. Modifiez le fichier de contraintes pour connecter vos entrées / sorties du système avec les broches de la carte. Réglez l’horloge pour que sa fréquence soit à 100MHz.**

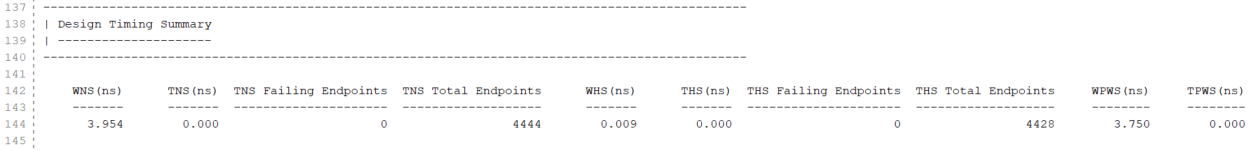
Une image contenant texte, Police, capture d’écran, algèbre

Description générée automatiquement

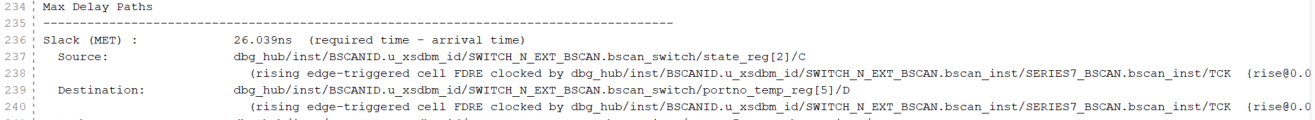
**11. Lancez l’implémentation puis étudiez le rapport de timing (vérifiez les violations de set up et de hold et identifiez le chemin critique).**

Le TNS (Total Negative Slack) est utilisé pour mesurer la marge de temporisation négative totale dans notre conception. Un TNS négatif signifie que certaines contraintes de temporisation ne sont pas respectées. Dans notre cas, nous avons observé un TNS de 0.000ns, ce qui indique que toutes les contraintes de temporisation sont satisfaites pour tous les points évalués.

De même, le THS (Total Hold Slack) est utilisé pour mesurer la marge de temporisation de maintien totale dans notre conception. Un THS négatif indiquerait un risque de violation des contraintes de maintien. Dans notre cas, nous avons constaté un THS de 0.000ns, ce qui indique que toutes les contraintes de maintien sont également respectées pour tous les points évalués.



Chemin critique :



**Resultats de la ILA**

Les LEDs sont pilotées à l'aide de machines à états. Pour tester ce système sur la carte, nous avons défini un nombre de cycles fixe à 4. Le nombre de coups d'horloge est fixé à 200 millions, correspondant à une fréquence du système de 100 MHz. Le compteur de cycle est représenté par son signal interne "Clk\_counter".

La LED change d'état toutes les deux secondes, soit une fois par cycle de "Clk\_counter". Ainsi, la LED clignote deux fois pour atteindre le nombre de cycles défini.

Sur les captures d'écran ci-dessous, on peut voir que le système passe à l'état suivant uniquement lorsque "Clk\_counter" atteint sa valeur maximale de 4. Dans ce cas, le système passe de son état initial à l'état rouge après 2 clignotements de la LED en blanc.

Une image contenant capture d’écran, texte, logiciel, Logiciel multimédia

Description générée automatiquement

Cependant, si le système est dans l'état rouge et que "Clk\_counter" atteint 4, il passe à l'état suivant en passant à l'état bleu. De même, si le système est dans l'état vert et que "Clk\_counter" atteint 4, il passe à l'état suivant en passant à l'état rouge.

Une image contenant capture d’écran, texte, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant capture d’écran, logiciel, Logiciel multimédia, texte

Description générée automatiquement

Une image contenant capture d’écran, texte, logiciel, Logiciel multimédia

Description générée automatiquement

**Test du bouton Restart :**

Dans ce paragraphe, nous testons le système en appuyant sur le bouton "Restart". Lorsque le bouton est enfoncé, l'état du système revient à l'état initial dans lequel il se trouvait au démarrage, et le compteur de cycle continue à incrémenter puis réinitialiser à zéro au moment où "Restart" est égale à 1. Ensuite, le système passe à l'état "Rouge" après deux clignotements de la LED en blanc (rouge, vert et bleu actifs en même temps).

Dans l'exemple ci-dessous, l'état du système était "bleu" avec la LED bleu allumée. Lorsque nous avons appuyé sur le bouton "Restart", l'état du système est revenu à l'état initial "idle" comme le montre la figure suivante avec la LED RGB allumée en blanc (c'est-à-dire avec les valeurs OUT\_R=1, OUT\_B=1 et OUT\_V=1). Le compteur de cycle a été réinitialisé à zéro lorsque la variable "Restart" est égale à 1.

Une image contenant capture d’écran, logiciel, Logiciel multimédia, texte

Description générée automatiquement

Cela s'applique également lorsque l'état du système est rouge, vert ou à l'état initial.

**Test du bouton Reset :**

Lorsqu'on appuie sur le bouton "Reset", tout le système est réinitialisé à zéro, y compris le compteur de cycle et les LED qui s'éteignent puis le système revient à son état initial tel qu'il était au moment du démarrage ensuite passe à l'état "Rouge" après deux clignotements de la LED en blanc.

Une image contenant capture d’écran, texte, logiciel, Logiciel multimédia

Description générée automatiquement