PROJET 4A

Compte rendu de Projet 4A

Communication entre FPGA et Arduino

MALYANI Youssef
AKIAOU Ayoub
ElSAYED Raif

encadré par : M. J.Duboisl

04 Janvier 2022

Sommaire

Liste des figures

1 INTRODUCTION :

Notre projet a pour but d'utiliser un signal PWM de type factoriel pour contrôler la vitesse d'un moteur à courant continu (CC) en mettant en pratique l'accélération matérielle en utilisant un composant basé sur FPGA pour calculer la factorielle d'un signal, afin d'accélérer les calculs en utilisant la puissance des FPGA et un Arduino UNO R3 pour bien gérer notre moteur et assurer la communication entre l'Arduino et le FPGA.

2 PLAN DE TRAVAIL :

Le projet est divisé en plusieurs étapes pour assurer une réalisation efficace et une compréhension claire du projet. La première étape vise à comprendre les détails du projet et à installer les différents composants électroniques nécessaires, la deuxième étape consiste à construire et programmer nos composants et la dernière étape consiste à vérifier le bon fonctionnement de notre moteur.
Remarque : Ce projet est un travail en trinôme ou nous avons travaillée simultanément pour bien gérer toutes les étapes de ce projet

2.1 PREMIER PARTIE : Compréhension du projet.

L'idée principale du projet ? Notre projet consiste à utiliser un microcontrôleur Arduino pour envoyer des commandes de mouvement à notre moteur, tandis que le FPGA est utilisé pour contrôler la vitesse et le sens de rotation du moteur en utilisant un algorithme de commande factorielle

Pourquoi utiliser L'algorithme de commande factorielle ? L'algorithme de commande factorielle est utilisé comme un régulateur de vitesse pour maintenir une vitesse constante du moteur en répondant à toutes les perturbations externes, telles que les variations de tension d'alimentation.

Le rôle principal de FPGA ? Le FPGA ici permet d'avoir une grande flexibilité et une meilleure performance de contrôle par rapport à l'utilisation d'un microcontrôleur seul, car il peut être reprogrammé pour s'adapter à différents types de systèmes de commande de moteur.

Communication Arduino FPGA ? Le but de la communication et que l'Arduino soit capable d'envoyer et de récupérer des données vers et depuis le FPGA en utilisant un protocole de communication tel que SPI ou UART ou bien créer un protocole (qui est notre cas). Une fois les données sont envoyées le FPGA peut ensuite utiliser ces données pour contrôler la vitesse de notre moteur

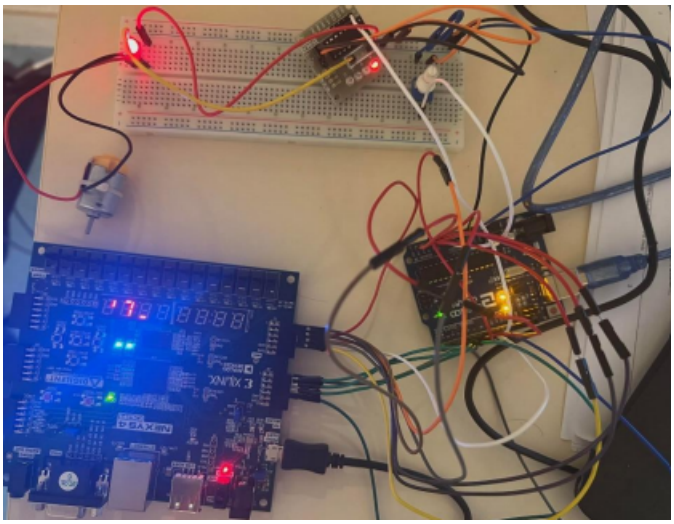

Figure 1: Schéma de notre projet

2.2 DEUXIEME PARTIE : Réalisation

2.2.1 Programmation de L'Arduino :

Le programme d'Arduino définit plusieurs broches d'entrée et de sortie, notamment une broche de sortie pour un signal PWM pour contrôler le moteur, et des broches d'entrée pour lire des données de l'FPGA et des broches de sortie pour envoyer des données à l'FPGA. Dans la fonction de configuration, on définit les modes de broche pour chaque broche d'entrée et de sortie et ouvre également un port de communication série à une vitesse de 9600 bauds. Dans la fonction de boucle, le programme lit les données de l'FPGA à l'aide de la fonction ReadFromFPGA (), génère un signal PWM à l'aide de la fonction analogWrite (), lit les données d'un potentiomètre à l'aide de la fonction analogRead () et envoie des données à l'FPGA à l'aide de la fonction WriteToFPGA (). Enfin, le programme affiche les données de l'FPGA et du potentiomètre sur le port série et attend 1 seconde avant de répéter la boucle.

```
Serial.print("\n Data FPGA = " );
Serial.print(val);
Serial.print("\n Data Poto = " );
Serial.print(val_reel);
```

Figure 2: Affichage de Data FPGA et Poto

```
int ReadFromFPGA()
{
    int val_tmp=0;

    // Bit 0
    val_0 = analogRead(From_FPGA_pin_1);
    if (val_0 > 512)
    {
        val_0 =1;
    }
    else
    {
        val_0=0;
    }
```

Figure 3: Fonction ReadFromFPGA.

```
void WriteToFPGA(int val_reel_tmp)
{
    if (val_reel_tmp==0){
        digitalWrite(To_FPGA_pin_1, LOW);       // 00000
        digitalWrite(To_FPGA_pin_2, LOW);
        digitalWrite(To_FPGA_pin_3, LOW);
        digitalWrite(To_FPGA_pin_4, LOW);
        digitalWrite(To_FPGA_pin_5, LOW);
        }
```

Figure 4: Fonction WriteToFPGA.

2.2.2 Programmation de FPGA :

Pour le code FPGA nous avons créé une entité appelée "FactorielMem" qui calcule la factorielle d'un nombre afin de gérer notre signale PWM.
On commence à vous présenter les entrées et les sorties de notre circuit :

- "Rst" est une entrée de réinitialisation qui, lorsqu'elle est mise au niveau haut, met la sortie à une valeur prédéfinie.
- L'entrée "Clk" est l'entrée d'horloge qui déclenche le calcul de la factorielle.
- L'entrée "D_in" est une entrée de 5 bits qui contient le nombre dont la factorielle doit être calculée.
- La sortie "D_out_Ready" est un signal qui indique quand le calcul de la factorielle est terminé et que la sortie est prête.
- La sortie "D_out" est une sortie 8 bits qui contient le résultat du calcul de la factorielle.
- La sortie "D_out_Arduino" est similaire à la sortie "D_out" mais elle est destinée à la communication avec un Arduino.

Dans la partie “architecture” un composant “Factoriel” est appelé pour calculer la factorielle d'un nombre qui possède 5 ports d'entrée et de sortie qui sont mis en correspondance avec les ports d'entrée et de sortie du composant FactorielMem (Notre composent top).
Notre code comprend également quelques signaux :

- "D_outReady_tmp", "D_out_tmp" et "D_in_tmp". "D_outReady_tmp" est un signal qui contient l'état de disponibilité de la sortie du composant FactorielN.
- "D_out_tmp" est un signal qui contient la sortie du composant FactorielN.
- "D_in_tmp" est un signal qui contient l'entrée du composant FactorielN.

En plus il existe un processus appelé "P_data_out", qui contrôle la sortie du composant FactorielMem. Le processus commence par vérifier si l'entrée de réinitialisation (Rst) est élevée. Si c'est le cas, la sortie est fixée à une valeur prédéfinie (1 pour D_out (0) et 0 pour D_out (P-1 downto 1)).
Si l'entrée de réinitialisation est basse, le processus continue à vérifier l'entrée d'horloge. Si l'entrée d'horloge est haute et que c'est un front montant, le processus vérifie la valeur d'entrée. Si la valeur d'entrée est nulle, le processus définit l'entrée du composant FactorielN à 1. Si la valeur d'entrée n'est pas nulle, le processus définit l'entrée du composant FactorielN à la valeur d'entrée. Ensuite, le processus vérifie le signal de sortie prêt du composant FactorielN. S'il est élevé, le processus vérifie la sortie du composant FactorielN et s'il est supérieur à la valeur prédéfinie (256 dans ce cas), la sortie du composant FactorielMem est fixée à "FF" (255 en décimal), sinon, la sortie du composant FactorielMem est fixée à la sortie du composant FactorielN.
Enfin, la sortie "D_out_Ready" est réglée sur le signal de sortie prête du composant FactorielN.
On attire votre attention que notre code a la capacité de gérer la saturation de la sortie et le cas particulier où l'entrée est 0 et Il a également la capacité d'assurer la communication avec l'Arduino via la sortie "D_out_Arduino".

```
FAC5: FactorielN PORT MAP(
        Rst => Rst,
        Clk => Clk,
        D_in => D_in_tmp,
        D_out_Ready => D_outReady_tmp,
        D_out => D_out_tmp
);
```

Figure 5: Component Factoriel.

```
P_data_out : process(Rst, Clk, D_outReady_tmp)
begin
        if Rst ='1' then
                D_out(P-1 downto 1) <=(others =>'0');
                D_out(0) <= '1';
        elsif Clk = '1' and Clk'event then
                -- 0!=1
                if D_in=b"00000" then
                        D_in_tmp <= b"00001";
                else
                        D_in_tmp<= D_in;
                end if;
                -- Saturation de la sortie
                if D_outReady_tmp='1' then
                        if D_out_tmp > x"00000000000000000000000008" & b"00000" then  -- 256
                                D_out <= x"FF";
                                D_out_Arduino <= x"FF";
                        else
                                D_out <= D_out_tmp(P-1 downto 0);
                                D_out_Arduino <= D_out_tmp(P-1 downto 0);
                        end if;
                end if;
        end if;
end process;
D_out_Ready <=D_outReady_tmp;
```

Figure 6: Process du code Principale

2.2.3 TROISIEME PARTIE : Résultats.

Sur le Moniteur série peut bien constater que que avant l'intégration de FPGA on un valeur presque fixe des signaux qui arrivent de la carte FPGA, et Après l'intégration du FPGA on peut voir que la valeur change quand on bouge le potentiomètre.

```
Data Poto = 31.00
Data FPGA = 26
Data Poto = 31.00
Data FPGA = 26
Data Poto = 31.00
Data FPGA = 25
Data Poto = 31.00
Data FPGA = 24
Data Poto = 31.00
```

Figure 7: Avant l'intégration de FPGA.

```
Data Poto = 14.00
Data FPGA = 168
Data Poto = 25.00
Data FPGA = 168
Data Poto = 30.00
Data FPGA = 188
Data Poto = 31.00
Data FPGA = 63
Data Poto = 30.00
Data FPGA = 187
Data Poto = 30.00
Data FPGA = 185
Data Poto = 30.00
Data FPGA = 184
```

Figure 8: Après l'intégration de FPGA.

2.3 PROBLÈMES RENCONTRER :

Dans ce projet, Au niveau du protocole, en premier on a essayé de travailler selon le protocole SPI mais on a trouvé que c'était un peu compliqué pour cela on a créé notre propre protocole.
Au niveau de fonctionnement on peut voir très bien que notre protocole fonctionne et on peut bien varier la vitesse de notre moteur mais le seul doute rencontrer et au bout de la valeur du potentiomètre pur une valeur de 0 notre moteur arrête de tourner mais après quelque instant il redémarre de nouveau on arrive pas bien à savoir notre erreur arrive d'où ?

3 Conclusion.

Ce fut une expérience d'apprentissage enrichissante pour nous de travailler sur ce projet. La problématique nous a permis de faire appel à des connaissances théoriques et pratiques variées qui sont souvent utilisées dans le domaine de Systèmes Embarques afin de bien gérer ce projet.

ANNEX

- PROGRAMME ARDUINO:

```
// SOrtie PWM
int To_Moteur_PWM = 9;

// Entree potentiometre
int Potent_pin = A0;
int val_Pot = 0;
float val_reel = 0.0;

// Sortie To FPGA
int To_FPGA_pin_1 = 2;
int To_FPGA_pin_2 = 3;
int To_FPGA_pin_3 = 4;
int To_FPGA_pin_4 = 5;
int To_FPGA_pin_5 = 10;

// Donnees from FPGA
int From_FPGA_pin_1 = A1;
int From_FPGA_pin_2 = A2;
int From_FPGA_pin_3 = A3;
int From_FPGA_pin_4 = A4;
int From_FPGA_pin_5 = A5;
int From_FPGA_pin_6 = A6;
int From_FPGA_pin_7 = A7;
int From_FPGA_pin_8 = 8;

int val = 0;

int val_0 = 0;
int val_1 = 0;
int val_2 = 0;
int val_3 = 0;
int val_4 = 0;
int val_5 = 0;
int val_6 = 0;
int val_7 = 0;

void setup()
{
        //
```

```
        pinMode(To_FPGA_pin_1, OUTPUT);
        pinMode(To_FPGA_pin_2, OUTPUT);
        pinMode(To_FPGA_pin_3, OUTPUT);
        pinMode(To_FPGA_pin_4, OUTPUT);
        pinMode(To_FPGA_pin_5, OUTPUT);
        pinMode(To_Moteur_PWM, OUTPUT);
        // entrees
      pinMode(From_FPGA_pin_1, INPUT);
      pinMode(From_FPGA_pin_2, INPUT);
      pinMode(From_FPGA_pin_3, INPUT);
      pinMode(From_FPGA_pin_4, INPUT);
      pinMode(From_FPGA_pin_5, INPUT);
      pinMode(From_FPGA_pin_6, INPUT);
      pinMode(From_FPGA_pin_7, INPUT);
      pinMode(From_FPGA_pin_8, INPUT);
        // Seria port
        Serial.begin(9600);
}

void loop()
{
        // Lecture entrees FPGA
        val=ReadFromFPGA();

        //Generation PWM/
        analogWrite(To_Moteur_PWM, (val+1)*2-1);

        //Lecture Pot /
        val_Pot = analogRead(Potent_pin);
        val_reel=floor(double(val_Pot)*31.0/1023.0);

        // Envoie 5 bits au FPGA /
        WriteToFPGA(val_reel);

        // Display data Serial port

        Serial.print("\n Data FPGA = " );
        Serial.print(val);
        Serial.print("\n Data Poto = " );
        Serial.print(val_reel);
                //Serial.print("\n MO = " );
 //       Serial.print();
        delay(1000);

        }
```

```
int ReadFromFPGA()
{
        int val_tmp=0;

        // Bit 0
        val_0 = analogRead(From_FPGA_pin_1);
        if (val_0 > 512)
        {
                val_0 =1;
        }
        else
        {
                val_0=0;
        }

        // Bit 1
        val_1 = analogRead(From_FPGA_pin_2);
        if (val_1 > 512)
        {
                val_1 =2;
        }
        else
        {
                val_1=0;
        }

        // Bit 2
        val_2 = analogRead(From_FPGA_pin_3);
        if (val_2 > 512)
        {
                val_2 =4;
        }
        else
        {
                val_2=0;
        }

        // Bit 3
        val_3 = analogRead(From_FPGA_pin_3);
        if (val_3 > 512)
        {
                val_3 =8;
        }
        else
        {
```

```
                val_2=0;
        }

        // Bit 4
        val_4 = analogRead(From_FPGA_pin_5);
        if (val_4 > 512)
        {
                val_4 =16;
        }
        else
        {
                val_4=0;
        }

        // Bit 5
        val_5 = analogRead(From_FPGA_pin_6);
        if (val_5 > 512)
        {
                val_5 =32;
        }
        else
        {
                val_5=0;
        }

        // Bit 6
        val_6 = analogRead(From_FPGA_pin_7);
        if (val_6 > 512)
        {
                val_6 =64;
        }
        else
        {
                val_6=0;
        }
// Bit 7
        val_7 = analogRead(From_FPGA_pin_8);
        if (val_7 > 512)
        {
                val_7 =128;
        }
        else
        {
                val_7=0;
        }
```

```
        // Resultat
        val_tmp = val_0+val_1+val_2+val_3+val_4+val_5+val_6;+val_7;

        // Return
        return val_tmp;
}

void WriteToFPGA(int val_reel_tmp)
{
        if (val_reel_tmp==0){
                digitalWrite(To_FPGA_pin_1, LOW);          // 00000
                digitalWrite(To_FPGA_pin_2, LOW);
                digitalWrite(To_FPGA_pin_3, LOW);
                digitalWrite(To_FPGA_pin_4, LOW);
                digitalWrite(To_FPGA_pin_5, LOW);
                }
        else if (val_reel_tmp==1){
                digitalWrite(To_FPGA_pin_1, HIGH);
// 00001
                digitalWrite(To_FPGA_pin_2, LOW);
                digitalWrite(To_FPGA_pin_3, LOW);
                digitalWrite(To_FPGA_pin_4, LOW);
                digitalWrite(To_FPGA_pin_5, LOW);   }
        else if (val_reel_tmp==2){
                digitalWrite(To_FPGA_pin_1, LOW);          // 00010
                digitalWrite(To_FPGA_pin_2, HIGH);
                digitalWrite(To_FPGA_pin_3, LOW);
                digitalWrite(To_FPGA_pin_4, LOW);
                digitalWrite(To_FPGA_pin_5, LOW); }
        else if (val_reel_tmp==3){
                digitalWrite(To_FPGA_pin_1, HIGH);
// 00011
                digitalWrite(To_FPGA_pin_2, HIGH);
                digitalWrite(To_FPGA_pin_3, LOW);
                digitalWrite(To_FPGA_pin_4, LOW);
                digitalWrite(To_FPGA_pin_5, LOW);
                }
        else if (val_reel_tmp==4){
                digitalWrite(To_FPGA_pin_1, LOW);          // 00100
                digitalWrite(To_FPGA_pin_2, LOW);
                digitalWrite(To_FPGA_pin_3, HIGH);
                                digitalWrite(To_FPGA_pin_4, LOW);
                digitalWrite(To_FPGA_pin_5, LOW); }

        else if (val_reel_tmp==5){
```

```
                digitalWrite(To_FPGA_pin_1, HIGH);
// 00101
                digitalWrite(To_FPGA_pin_2, LOW);
                digitalWrite(To_FPGA_pin_3, HIGH);
                                digitalWrite(To_FPGA_pin_4, LOW);
                digitalWrite(To_FPGA_pin_5, LOW); }
        else if (val_reel_tmp==6){
                digitalWrite(To_FPGA_pin_1, LOW);          // 00110
                digitalWrite(To_FPGA_pin_2, HIGH);
                digitalWrite(To_FPGA_pin_3, HIGH);
                digitalWrite(To_FPGA_pin_4, LOW);
                digitalWrite(To_FPGA_pin_5, LOW); }
        else if (val_reel_tmp==7){
                digitalWrite(To_FPGA_pin_1, HIGH);
// 00111
                digitalWrite(To_FPGA_pin_2, HIGH);
                digitalWrite(To_FPGA_pin_3, HIGH);
                digitalWrite(To_FPGA_pin_4, LOW);
                digitalWrite(To_FPGA_pin_5, LOW); }
        else if (val_reel_tmp==8){
                digitalWrite(To_FPGA_pin_1, LOW);          // 01000
                digitalWrite(To_FPGA_pin_2, LOW);
                digitalWrite(To_FPGA_pin_3, LOW);
                digitalWrite(To_FPGA_pin_4, HIGH);
                digitalWrite(To_FPGA_pin_5, LOW); }
        else if (val_reel_tmp==9){
                digitalWrite(To_FPGA_pin_1, HIGH);
// 01001
                digitalWrite(To_FPGA_pin_2, LOW);
                digitalWrite(To_FPGA_pin_3, LOW);
                digitalWrite(To_FPGA_pin_4, HIGH);
                digitalWrite(To_FPGA_pin_5, LOW); }
        else if (val_reel_tmp==10){
                digitalWrite(To_FPGA_pin_1, LOW);          // 01010
                digitalWrite(To_FPGA_pin_2, HIGH);
                digitalWrite(To_FPGA_pin_3, LOW);
                digitalWrite(To_FPGA_pin_4, HIGH);
                digitalWrite(To_FPGA_pin_5, LOW); }
        else if (val_reel_tmp==11){
                digitalWrite(To_FPGA_pin_1, HIGH);
// 01011
                digitalWrite(To_FPGA_pin_2, HIGH);
                digitalWrite(To_FPGA_pin_3, LOW);
                digitalWrite(To_FPGA_pin_4, HIGH);
                digitalWrite(To_FPGA_pin_5, LOW); }
        else if (val_reel_tmp==12){
```

```
                digitalWrite(To_FPGA_pin_1, LOW);          // 01100
                digitalWrite(To_FPGA_pin_2, LOW);

                digitalWrite(To_FPGA_pin_3, HIGH);
                digitalWrite(To_FPGA_pin_4, HIGH);
                digitalWrite(To_FPGA_pin_5, LOW); }
        else if (val_reel_tmp==13){
                digitalWrite(To_FPGA_pin_1, HIGH);
// 01101
                digitalWrite(To_FPGA_pin_2, LOW);
                digitalWrite(To_FPGA_pin_3, HIGH);
                digitalWrite(To_FPGA_pin_4, HIGH);
                digitalWrite(To_FPGA_pin_5, LOW); }
        else if (val_reel_tmp==14){
                digitalWrite(To_FPGA_pin_1, LOW);          // 01110
                digitalWrite(To_FPGA_pin_2, HIGH);
                digitalWrite(To_FPGA_pin_3, HIGH);
                digitalWrite(To_FPGA_pin_4, HIGH);
                digitalWrite(To_FPGA_pin_5, LOW); }
        else if (val_reel_tmp==15){
                digitalWrite(To_FPGA_pin_1, HIGH);
// 01111
                digitalWrite(To_FPGA_pin_2, HIGH);
                digitalWrite(To_FPGA_pin_3, HIGH);
                digitalWrite(To_FPGA_pin_4, HIGH);
                digitalWrite(To_FPGA_pin_5, LOW); }
          else if (val_reel_tmp==16){
                digitalWrite(To_FPGA_pin_1, LOW);          // 10000
                digitalWrite(To_FPGA_pin_2, LOW);
                digitalWrite(To_FPGA_pin_3, LOW);
                digitalWrite(To_FPGA_pin_4, LOW);
                digitalWrite(To_FPGA_pin_5, HIGH); }
        else if (val_reel_tmp==17){
                digitalWrite(To_FPGA_pin_1, HIGH);
// 10001
                digitalWrite(To_FPGA_pin_2, LOW);
                digitalWrite(To_FPGA_pin_3, LOW);
                digitalWrite(To_FPGA_pin_4, LOW);
                digitalWrite(To_FPGA_pin_5, HIGH);
                }
        else if (val_reel_tmp==18){
                digitalWrite(To_FPGA_pin_1, LOW);          // 10010
                digitalWrite(To_FPGA_pin_2, HIGH);
                digitalWrite(To_FPGA_pin_3, LOW);
                digitalWrite(To_FPGA_pin_4, LOW);
                digitalWrite(To_FPGA_pin_5, HIGH); }
```

```
        else if (val_reel_tmp==19){
                digitalWrite(To_FPGA_pin_1, HIGH);
// 10011
                digitalWrite(To_FPGA_pin_2, HIGH);
                digitalWrite(To_FPGA_pin_3, LOW);
                digitalWrite(To_FPGA_pin_4, LOW);
                digitalWrite(To_FPGA_pin_5, HIGH); }
        else if (val_reel_tmp==20){
                digitalWrite(To_FPGA_pin_1, LOW);          // 10100
                digitalWrite(To_FPGA_pin_2, LOW);
                digitalWrite(To_FPGA_pin_3, HIGH);
                digitalWrite(To_FPGA_pin_4, LOW);
                digitalWrite(To_FPGA_pin_5, HIGH); }
        else if (val_reel_tmp==21){
                digitalWrite(To_FPGA_pin_1, HIGH);
// 10101
                digitalWrite(To_FPGA_pin_2, LOW);
                digitalWrite(To_FPGA_pin_3, HIGH);
                digitalWrite(To_FPGA_pin_4, LOW);
                digitalWrite(To_FPGA_pin_5, HIGH); }
        else if (val_reel_tmp==22){
                digitalWrite(To_FPGA_pin_1, LOW);          // 10110
                digitalWrite(To_FPGA_pin_2, HIGH);
                digitalWrite(To_FPGA_pin_3, HIGH);
                digitalWrite(To_FPGA_pin_4, LOW);
                digitalWrite(To_FPGA_pin_5, HIGH); }
        else if (val_reel_tmp==23){
                digitalWrite(To_FPGA_pin_1, HIGH);
// 10111
                digitalWrite(To_FPGA_pin_2, HIGH);
                digitalWrite(To_FPGA_pin_3, HIGH);
                digitalWrite(To_FPGA_pin_4, LOW);
                digitalWrite(To_FPGA_pin_5, HIGH); }
        else if (val_reel_tmp==24){
                digitalWrite(To_FPGA_pin_1, LOW);          // 11000
                digitalWrite(To_FPGA_pin_2, LOW);
                digitalWrite(To_FPGA_pin_3, LOW);
                digitalWrite(To_FPGA_pin_4, HIGH);
                digitalWrite(To_FPGA_pin_5, HIGH); }

        else if (val_reel_tmp==25){
                digitalWrite(To_FPGA_pin_1, HIGH);
// 11001
                digitalWrite(To_FPGA_pin_2, LOW);
                digitalWrite(To_FPGA_pin_3, LOW);
                digitalWrite(To_FPGA_pin_4, HIGH);
```

```
                digitalWrite(To_FPGA_pin_5, HIGH); }
        else if (val_reel_tmp==26){
                digitalWrite(To_FPGA_pin_1, LOW);          // 11010

                digitalWrite(To_FPGA_pin_2, HIGH);
                digitalWrite(To_FPGA_pin_3, LOW);
                digitalWrite(To_FPGA_pin_4, HIGH);
                digitalWrite(To_FPGA_pin_5, HIGH); }
        else if (val_reel_tmp==27){
                digitalWrite(To_FPGA_pin_1, HIGH);
// 11011
                digitalWrite(To_FPGA_pin_2, HIGH);
                digitalWrite(To_FPGA_pin_3, LOW);
                digitalWrite(To_FPGA_pin_4, HIGH);
                digitalWrite(To_FPGA_pin_5, HIGH); }
        else if (val_reel_tmp==28){
                digitalWrite(To_FPGA_pin_1, LOW);          // 11100
                digitalWrite(To_FPGA_pin_2, LOW);
                digitalWrite(To_FPGA_pin_3, HIGH);
                digitalWrite(To_FPGA_pin_4, HIGH);
                digitalWrite(To_FPGA_pin_5, HIGH); }
        else if (val_reel_tmp==29){
                digitalWrite(To_FPGA_pin_1, HIGH);
// 11101
                digitalWrite(To_FPGA_pin_2, LOW);
                digitalWrite(To_FPGA_pin_3, HIGH);
                digitalWrite(To_FPGA_pin_4, HIGH);
                digitalWrite(To_FPGA_pin_5, HIGH); }
        else if (val_reel_tmp==30){
                digitalWrite(To_FPGA_pin_1, LOW);          // 11110
                digitalWrite(To_FPGA_pin_2, HIGH);
                digitalWrite(To_FPGA_pin_3, HIGH);
                digitalWrite(To_FPGA_pin_4, HIGH);
                digitalWrite(To_FPGA_pin_5, HIGH); }
        else  {
                digitalWrite(To_FPGA_pin_1, HIGH);
// 11111
                digitalWrite(To_FPGA_pin_2, HIGH);
                digitalWrite(To_FPGA_pin_3, HIGH);
                digitalWrite(To_FPGA_pin_4, HIGH);
                digitalWrite(To_FPGA_pin_5, HIGH); }

}
```

- PROGRAMME FPGA:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-----------------------------------
entity FactorielMem is
 Generic ( N : positive := 108;
           M : positive := 5;
           P : positive := 8 );
-----------------------------------
Port (
 Rst: in  STD_LOGIC;
 Clk : in  STD_LOGIC;
 D_in : in  STD_LOGIC_VECTOR (M-1 downto 0);
 D_out_Ready: out  STD_LOGIC;
 D_out: out  STD_LOGIC_VECTOR (P-1 downto 0);
 D_out_Arduino: out  STD_LOGIC_VECTOR (P-1 downto 0)  );
end FactorielMem;
-------------------------------------------------------------
-------------------------------------------------------------
architecture Behavioral of FactorielMem is
-- signaux
signal D_outReady_tmp :STD_LOGIC :='0';
signal D_out_tmp :STD_LOGIC_VECTOR
(N+M-1 downto 0):=x"00000000000000000000000000" & b"00001";
signal D_in_tmp  :STD_LOGIC_VECTOR (M-1 downto 0):=b"00000";
---------- appelle de composant
COMPONENT FactorielN
Generic ( N : positive := 108;  M : positive := 5);
PORT(    Rst : IN std_logic;
         Clk : IN std_logic;
         D_in : IN std_logic_vector(M-1 downto 0);
         D_out_Ready : OUT std_logic;
         D_out : OUT std_logic_vector(N+M-1 downto 0));
END COMPONENT;
-------------
begin
 FAC5: FactorielN generic map( N => 108, M =>5)
 PORT MAP(Rst => Rst,Clk => Clk,D_in => D_in_tmp,D_out_Ready
 => D_outReady_tmp,D_out => D_out_tmp);

-- Memorisation de la sortie + Gestion de 0!
P_data_out : process(Rst, Clk, D_outReady_tmp)
begin
if Rst ='1' then
```

```
D_out(P-1 downto 1) <=(others =>'0');
 D_out(0) <= '1';
 elsif Clk = '1' and Clk'event then
   -- 0!=1
if D_in=b"00000" then
 D_in_tmp <= b"00001";

 else
 D_in_tmp<= D_in;
 end if;
-- Saturation de la sortie
if D_outReady_tmp='1' then
if D_out_tmp > x"00000000000000000000000008" & b"00000"
then  -- 256
D_out <= x"FF";
 D_out_Arduino <= x"FF";
else
D_out <= D_out_tmp(P-1 downto 0);
D_out_Arduino <= D_out_tmp(P-1 downto 0);
end if;
end if;
end if;
end process;
D_out_Ready <=D_outReady_tmp;
end Behavioral;
```

- <u>COMPOSANT FactorielN:</u>

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_arith.ALL;
use IEEE.STD_LOGIC_signed.ALL;
```

```
entity FactorielN is
 Generic ( N : positive := 108;  M : positive := 5);
PORT(    Rst : IN std_logic;
         Clk : IN std_logic;
         D_in : IN std_logic_vector(M-1 downto 0);
         D_out_Ready : OUT std_logic;
         D_out : OUT std_logic_vector(N+M-1 downto 0));
end FactorielN;

architecture Behavioral of FactorielN is
```

```
signal fact : std_logic_vector(N+M-1 downto 0)
:= x"00000000000000000000000000" & b"00001";
signal fact1 : std_logic_vector(N+2*M-1 downto 0);
signal a : std_logic_vector(M-1 downto 0) := b"00000";
begin
P_factorial: process(Rst, Clk)
begin
    if Rst = '1' then
        fact <= x"00000000000000000000000000" & b"00001";
        a <= b"00000";
    elsif Clk = '1' and Clk'event then
        if a < D_in then
            a <= a + 1;
            fact1 <= (fact * a);
            fact<= fact1( N+M-1downto 0);
        else
            D_out_Ready <= '1';
            D_out <= fact;
        end if;
    end if;
end process;
end Behavioral;
```

- Pinout FPGA

```
## Clock signal
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 }.
[get_ports { Clk }]; #IO_L12P_T1_MRCC_35 Sch=clk100mhz
create_clock -add -name sys_clk_pin -period 12.00 -waveform {0 5}
[get_ports {Clk}];
##Switches

set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { Rst }]
## LEDs

set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 }
[get_ports { D_out[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 }
[get_ports { D_out[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 }
[get_ports { D_out[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 }
[get_ports { D_out[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 }
[get_ports { D_out[4] }]; #IO_L7P_T1_D09_14 Sch=led[4]
```

```
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 }
[get_ports { D_out[5] }]; #IO_L18N_T2_A11_D27_14 Sch=led[5]
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 }
[get_ports { D_out[6] }]; #IO_L17P_T2_A14_D30_14 Sch=led[6]
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 }
[get_ports { D_out[7] }]; #IO_L18P_T2_A12_D28_14 Sch=led[7]
##Pmod Header JC

set_property -dict { PACKAGE_PIN K1    IOSTANDARD LVCMOS33 }
[get_ports { D_out_Arduino[0] }]; #IO_L23N_T3_35 Sch=jc[1]
set_property -dict { PACKAGE_PIN F6    IOSTANDARD LVCMOS33 }
[get_ports { D_out_Arduino[1] }]; #IO_L19N_T3_VREF_35 Sch=jc[2]
set_property -dict { PACKAGE_PIN J2    IOSTANDARD LVCMOS33 }
[get_ports { D_out_Arduino[2] }]; #IO_L22N_T3_35 Sch=jc[3]
set_property -dict { PACKAGE_PIN G6    IOSTANDARD LVCMOS33 }
[get_ports { D_out_Arduino[3] }]; #IO_L19P_T3_35 Sch=jc[4]
set_property -dict { PACKAGE_PIN E7    IOSTANDARD LVCMOS33 }
[get_ports { D_out_Arduino[4] }]; #IO_L6P_T0_35 Sch=jc[7]
set_property -dict { PACKAGE_PIN J3    IOSTANDARD LVCMOS33 }
[get_ports { D_out_Arduino[5] }]; #IO_L22P_T3_35 Sch=jc[8]
set_property -dict { PACKAGE_PIN J4    IOSTANDARD LVCMOS33 }
[get_ports { D_out_Arduino[6] }]; #IO_L21P_T3_DQS_35 Sch=jc[9]
set_property -dict { PACKAGE_PIN E6    IOSTANDARD LVCMOS33 }
[get_ports {D_out_Arduino[7] }]; #IO_L5P_T0_AD13P_35 Sch=jc[10]


##Pmod Header JD

set_property -dict { PACKAGE_PIN H4    IOSTANDARD LVCMOS33 }
[get_ports { D_in[0] }]; #IO_L21N_T3_DQS_35 Sch=jd[1]
set_property -dict { PACKAGE_PIN H1    IOSTANDARD LVCMOS33 }
[get_ports { D_in[1] }]; #IO_L17P_T2_35 Sch=jd[2]
set_property -dict { PACKAGE_PIN G1    IOSTANDARD LVCMOS33 }
[get_ports { D_in[2] }]; #IO_L17N_T2_35 Sch=jd[3]
set_property -dict { PACKAGE_PIN G3    IOSTANDARD LVCMOS33 }
[get_ports {D_in[3]  }]; #IO_L20N_T3_35 Sch=jd[4]
set_property -dict { PACKAGE_PIN H2    IOSTANDARD LVCMOS33 }
[get_ports { D_in[4]  }]; #IO_L15P_T2_DQS_35 Sch=jd[7]
```