Travaux dirigés de Logique Programmable

Séance 2

L3 : 2020-2021

Rémi Griot

# Introduction

## Fichiers associés

Pour réaliser le TD, vous aurez besoin des documents suivants :

Les fichiers sources

* spimaster.vhd

Les test benchs associés :

* spimaster\_tb.vhd

## Résumé du TD2

Ce TD va vous permettre de revoir des notions du cours ainsi que de décrire une machine à état en VHDL

[1 Introduction 2](#_Toc55560896)

[2 Fréquence maximum d’un circuit logique 3](#_Toc55560897)

[3 Synchrone – Asynchrone / séquentiel - combinatoire 3](#_Toc55560898)

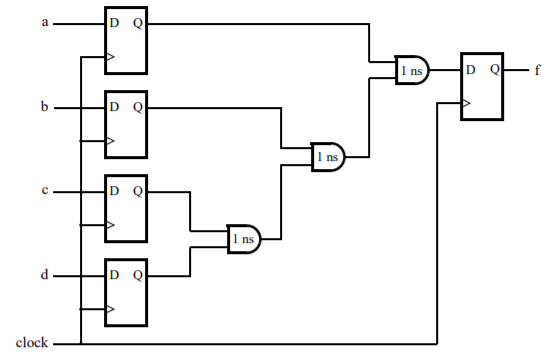
[4 Machine d’état SPI Master 4](#_Toc55560899)

# Fréquence maximum d’un circuit logique

* Calculer la fréquence maximum du circuit ci-dessous, sachant que l’on considère le skew comme négligeable, Tsu à 1ns et Tco à 1ns

Fmax= 1 / (Tco + Tpd +Tsu – skew)

1/(1\*10^-9 + 3 \* 1\*10^-9 + 1\*10^-9) = 1 / 5\*10^-9 = 200MHz



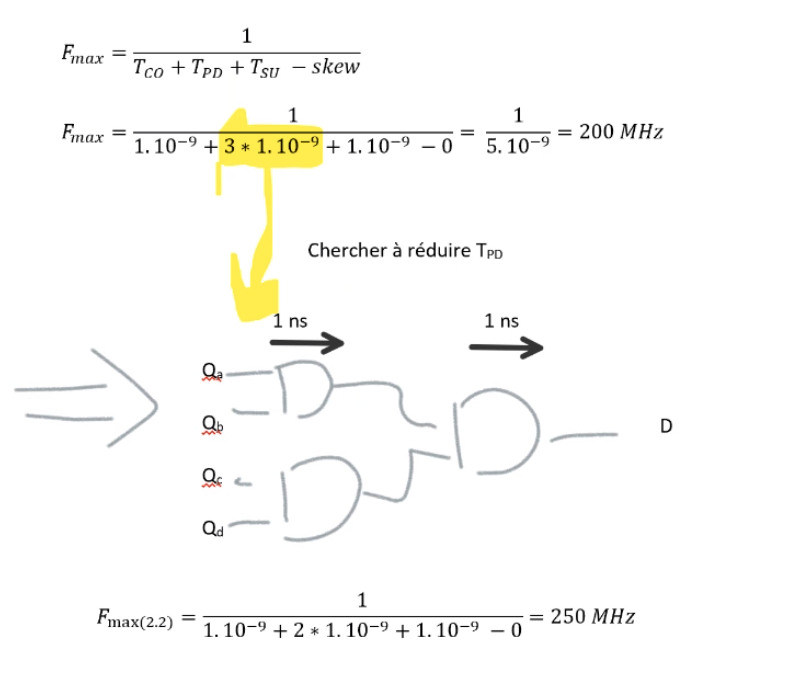
* Proposer un circuit équivalent fonctionnant à une fréquence maximum plus élevée. Quelle est cette fréquence ?

Tco = invariant, lié à la technologie

Tsu = invariant, lié à la technologie

On va donc modifier Tpd (éventuellement, pas toujours)

On va chercher à réduire Tpd

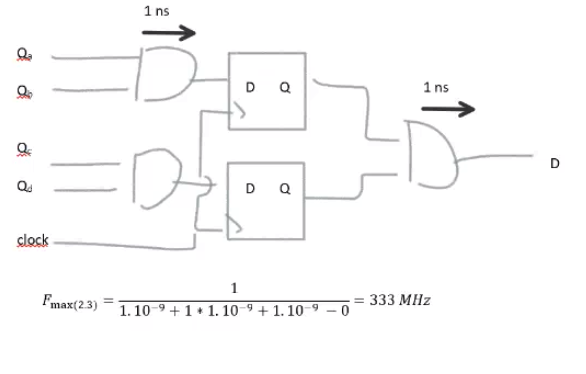


* Proposer une méthode qui, en modifiant le comportement du circuit mais en gardant l’équation logique, permettrai d’autoriser une fréquence maximum plus élevé.

Chercher de réduire le Tpd au minimum

Intégrer des étages de mémorisation pour limiter au maximum le temps de propagation

* Donc augmentation de la fréquence maximum (pas la fréquence totale attention)
* Au détriment de la consommation (ressources et énergies)



# Synchrone – Asynchrone / Séquentiel - Combinatoire

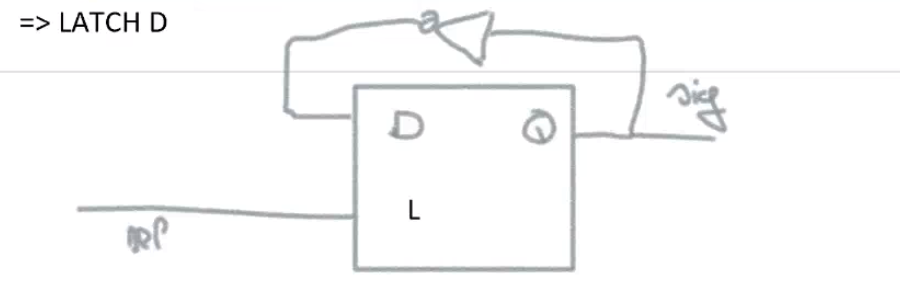
* Qualifier les circuits décrit ci-dessous. Dessiner le schéma associé quand il s’agit de description VHDL, et écrire la description en VHDL quand il s’agit de schéma

sig **<=** **not** sig **when** sel **=** '1' **else**

'0'**;**

**Asynchrone (pas d’horloge) et séquentiel (reboucle de la sortie vers l’entrée) -> (combinaison des deux) À éviter en FPGA**

* **LATCH D**

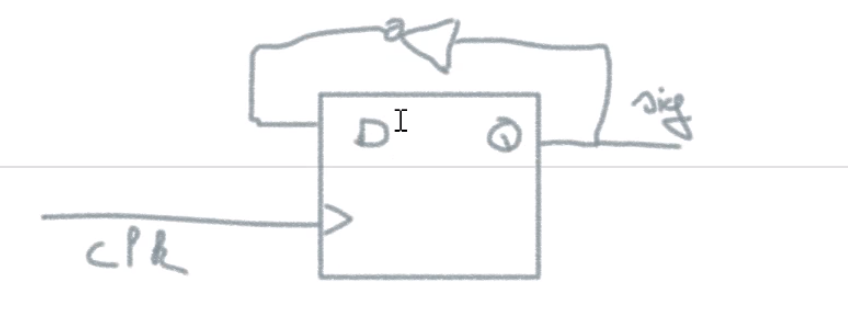


2)

sig **<=** **not** sig **when** rising\_edge (clk) ;

**Synchrone (horloge) et séquentiel (reboucle de la sortie vers l’entrée)**

* **Bascule D**



3)

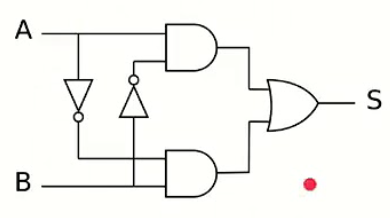
s **<=** a **or** b **or** c**;**

**Asynchrone et combinatoire (pas de rebouclage)**

* OU à 3 entrées



4)



**Asynchrone et combinatoire (pas de rebouclage)**

* S = A.not(B) + not(A).B = A xor B
* En VHDL: S <= A xor B;

5)

**process** **(**clk**,** rst**)**

**begin**

**if** rst **=** '1' **then**

dd **<=** '0'**;**

ddd **<=** '0'**;**

**elsif** **rising\_edge** **(**clk**)** **then**

d **<=** a + b**;**

dd **<=** d**;**

ddd **<=** dd**;**

**end** **if;**

**end** **process;**

**Synchrone et combinatoire (pas de rebouclage)**

* Additionneur avec 2 étages de mémorisation
* Pour du séquentiel on aurait eu quelque chose comme  
  d <= d + a + b ;

Une image contenant texte

Description générée automatiquement

# Machine d’état SPI Master

Nous allons étudier et décrire une machine à état synchrone sur CLK qui constitue un sous bloc d’un ensemble servant à transférer des données sur un bus SPI. Cette machine se compose de 4 états :

**Idle – Start – Data – Stop**

Les sorties de cette machine sont

* **CS** : Chip select (~Slave Select)
* **MOSI** : Master Output Slave Input

Les entrées de cette machine sont

* **CLK** : Horloge système
* **SEND** : Signal d’envoi de données
* **TICK\_SCLK** : Signal de synchro pour décaler le registre
* **DATA2SEND**: Donnée sur 8 bit transféré
* Ecrire l’entitée de la machine d’état (spimaster)

entity spimaster is

port(

CS: out std\_logic;

MOSI: out std\_logic;

CLK: in std\_logic;

SEND: in std\_logic;

TICK\_SCLK: in std\_logic;

DATA2SEND: in std\_logic\_vector(7 downto 0);

RST: in std\_logic

);

end entity spimaster;

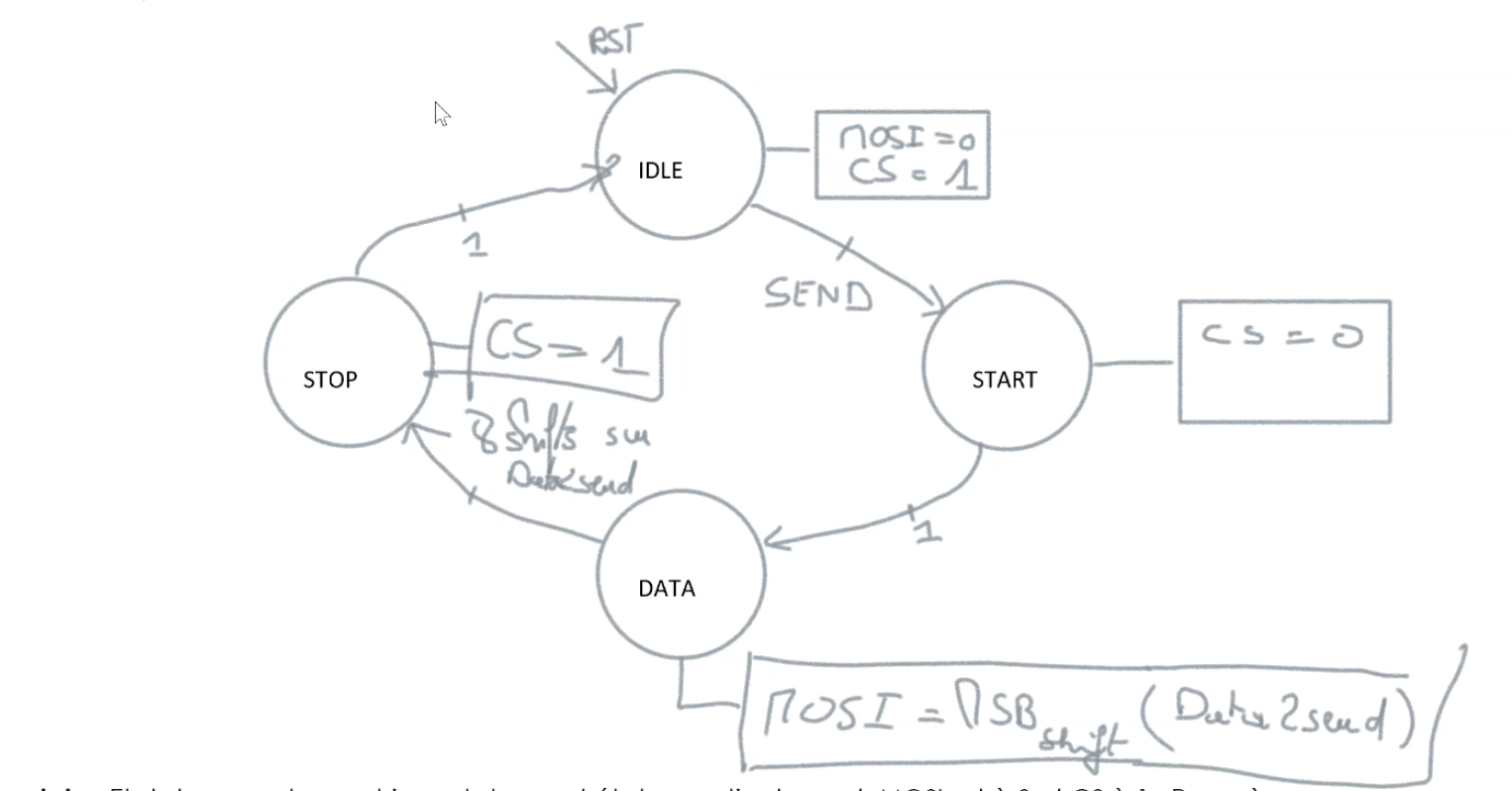
**Idle\_state** : Etat de repos, la machine est dans cet état en sortie de reset. MOSI est à 0 et CS à 1. Passe à Start si Send est à 1

**Start\_state** : Démarrage de la communication. CS est à 0. Passe sans condition à Data.

**Data\_state** : Décale le registre contenant la Data à envoyer (DATA2SEND) sur niveau haut de TICK\_SCLK. Passe à Stop quand les 8 bit du registre ont été décalé

**Stop\_state** : Arrêt de la communication. CS passe à 1.

* Dessiner l’évolution des états



* En vous basant sur spimaster.vhd, décrire en vhdl la machine d’état

-- pour visualiser les états sous EDA Playgroud  
mystate <= "00" when state = idle\_state else "01" when state = start\_state else "10" when state = data\_state else "11";

-- avant le begin de l’architecture ^

fsm\_p : process (clk, rst) is  
begin -- process fsm\_p  
if rst = '1' then  
cs <= '1';  
count <= (others => '0');  
shiftreg <= (others => '0');  
state <= idle\_state;  
  
elsif rising\_edge(clk) then  
case state is  
  
when idle\_state =>  
if send = '1' then  
state <= start\_state;  
shiftreg <= data2send;  
end if;  
  
when start\_state =>  
cs <= '0';  
state <= data\_state;  
  
when data\_state =>  
if tick\_sclk = '1' then  
if count < "111" then  
count <= count + '1';  
shiftreg <= shiftreg(6 downto 0) & '0';  
else  
state <= stop\_state;  
count <= (others => '0');  
end if;  
end if;  
when stop\_state =>  
cs <= '1';  
state <= idle\_state;  
when others =>  
state <= idle\_state;  
end case;  
end if;  
end process fsm\_p;

* Vérifier son fonctionnement avec spimaster\_tb.vhd

