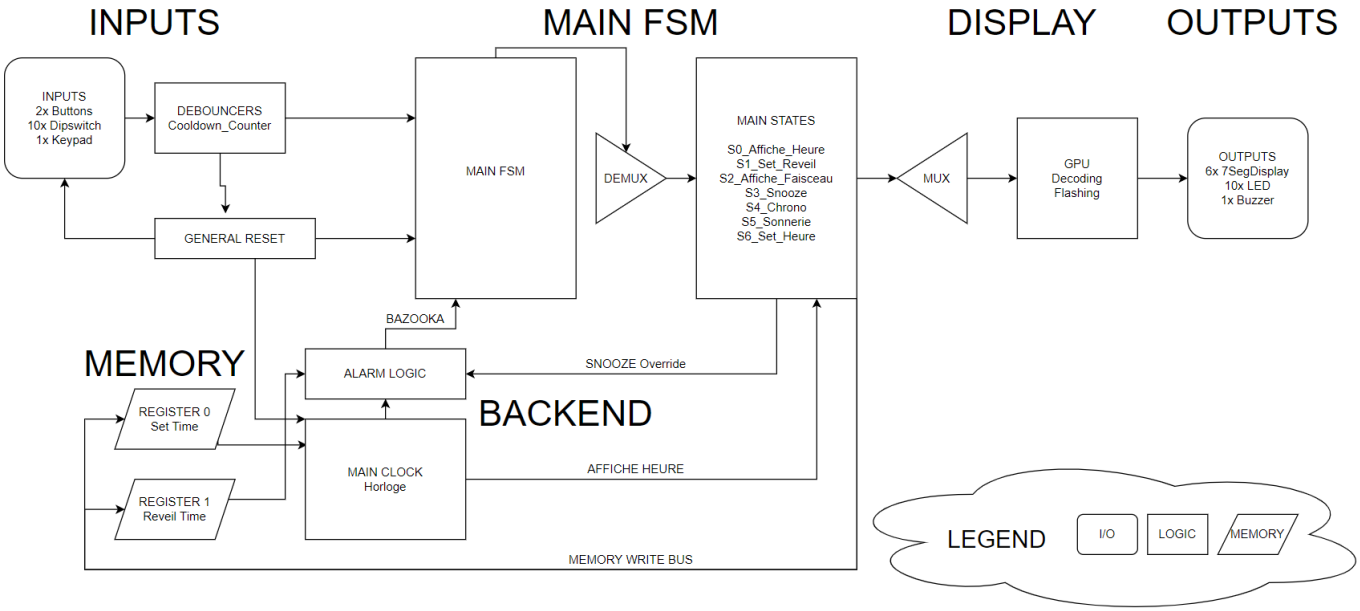
1. Description Générale du Projet



Notre projet est piloté par une FSM générale qui est traversée à l'aide des deux boutons de la FPGA. Nous avons un total de 7 états distincts, certains ont un affichage, d'autre un accès à la mémoire. Tout est abstrait et réutilisé, la mémoire est centralisée.

L'horloge et la logique d'alarme tournent en continu dans le BACKEND. Dès que l'heure courante est égalisée avec l'heure en mémoire dans le Registre R1 (réveil time), la logique d'alarme envoie un signal dit « BAZOOKA » qui a priorité dans la logique d'entrée de la FSM. Ce signal passe toujours la FSM dans son état « Sonnerie » peu importe l'état dans lequel la FSM se trouvait. Le « Snooze » démarre un compteur qui relance la sonnerie à travers le « BAZOOKA » une fois le compteur terminé.

2. Mode d'Emploi de l'Horloge Digitale

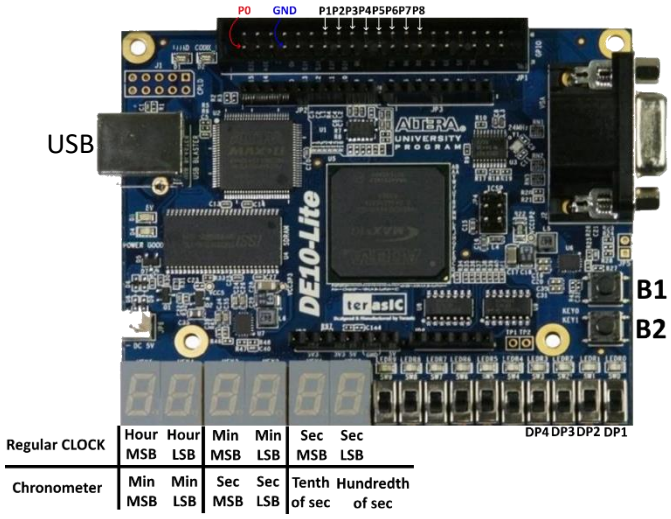
2.1. Mise en Place de l'Horloge

Brancher le Buzzer sur P0 et GND.
Brancher le Keypad sur P1 à P8.
Brancher la prise USB dans le port USB et dans l'ordinateur.
Basculer le DP1 pour effectuer un RESET.

2.2. Utilisation de l'Horloge

Les différents modes d'utilisation sont parcourus en appuyant sur B1 à multiples reprises, les voici :

$$Horloge \rightarrow_{B1} Chronomètre \rightarrow_{B1} Faisceau\ Horaire\ (DUBAI)$$
$$\uparrow\ B1$$



2.2.1. Chronomètre

Basculer DP3 pour redémarrer le chrono. Lever DP4 pour arrêter le chrono et puis baisser DP4 pour le redémarrer.

2.3. Réglages de l'Horloge

A partir de l'affichage de base, appuyer sur B2 pour accéder aux réglages et cycler entre les réglages à l'aide de B1, les voici :

$$Horloge \rightarrow_{B2} Régler\ Heure \rightarrow_{B1} Régler\ Réveil$$
$$\uparrow\ B1$$

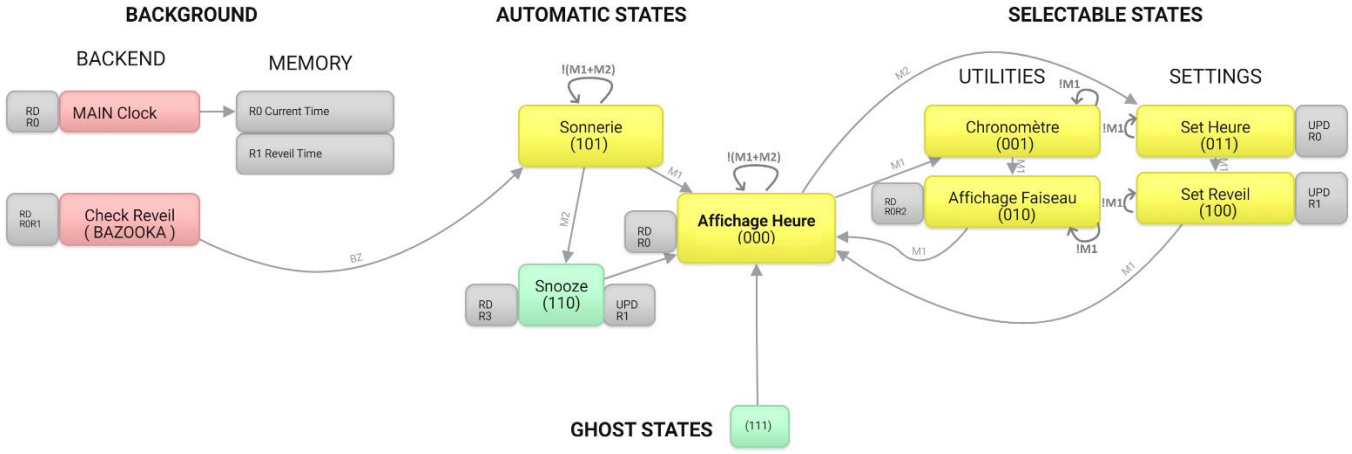
Pour régler une heure, renter une valeur sur le Keypad et la valider à l'aide de la touche « # ». La valeur en cours de modification clignote. Une fois les heures et minutes réglées, retourner au menu à travers B1.

2.4. Réveil

Lever DP2 pour activer le réveil. Une fois qu'il sonne, appuyez sur B1 pour l'éteindre ou appuyez sur B2 pour démarrer le « Snooze » qui durera 30 secondes (à but de test) et qui redémarrera la sonnerie à sa fin.

3. Solutions techniques

3.1. FSM Générale (FSM_Main)



L'état de base (IDLE) de la FSM est son état « Affichage Heure ».

De cet état, on accède à un **premier mode de fonctionnalités** grâce au bouton « M1 », celui-ci nous offre les fonctionnalités régulièrement utilisées par un utilisateur et passe de fonctionnalité à fonctionnalité à travers l'utilisation du bouton « M1 ».
Ceci libère le bouton « M2 » pour d'autres utilisations à l'intérieur de ces fonctionnalités de base.

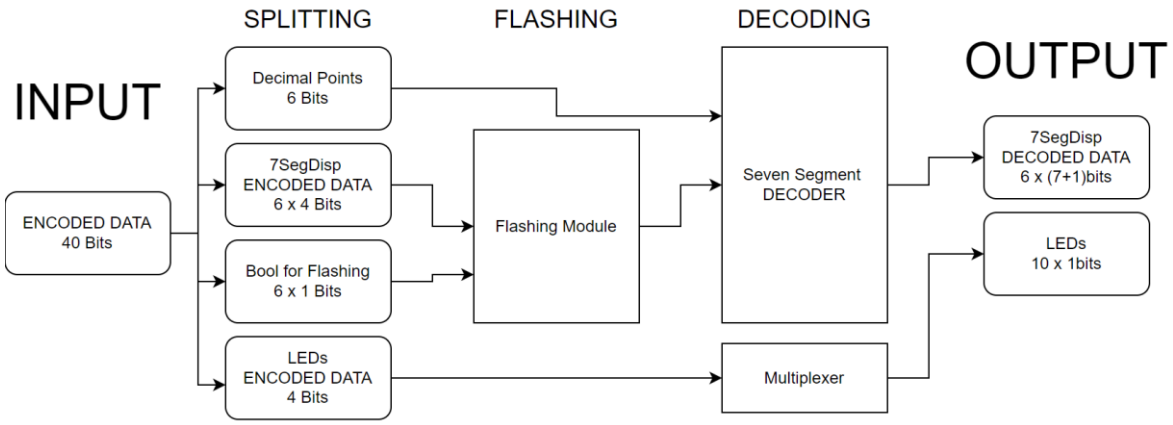
De l'état « Affiche Heure », on accède à un **second mode de réglages** à travers le bouton « M2 ». Ceci mobilise le bouton « M2 » mais nous le justifions car il n'est pas du tout pratique de devoir traverser tous les réglages à chaque fois qu'on souhaite lancer un chrono ou regarder un faisceau horaire.

Finalement, la sonnerie est déclenchée par de la logique interne à l'horloge et l'utilisateur peut soit retourner à « Affichage Heure » à travers « M1 » ou lancer le « Snooze » à travers « M2 ».

Table de Vérité de l'Implémentation de la FSM

	Q1	Q2	Q3	M1	M2	BZ	D1	D2	D3
Affiche Heure	0	0	0	0	0	0	0	0	0
	0	0	0	1	0	0	0	0	1
	0	0	0	.	1	0	0	1	1
Chrono	0	0	1	0	.	0	0	0	1
	0	0	1	1	.	0	0	1	0
Affiche Faiseau	0	1	0	0	.	0	0	1	0
	0	1	0	1	.	0	0	0	0
Set Heure	0	1	1	0	.	0	0	1	1
	0	1	1	1	.	0	1	0	0
Set Reveil	1	0	0	0	.	0	1	0	0
	1	0	0	1	.	0	0	0	0
Sonnerie	1	0	1	0	0	0	1	0	1
	1	0	1	0	1	0	1	1	0
	1	0	1	1	.	0	0	0	0
Snooze	1	1	0	.	.	0	0	0	0
False State	1	1	1	.	.	0	0	0	0
BAZOOKA	.	.	.	.	.	1	1	0	1

3.2. Module d'Affichage (GPU)

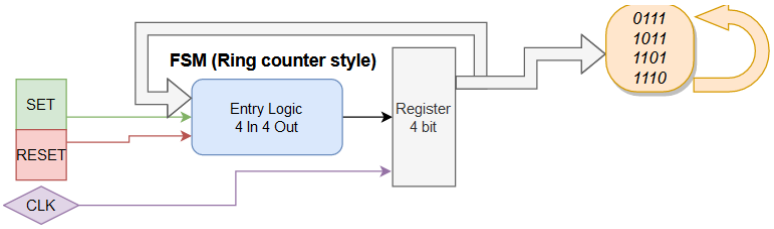


Pour maximiser les principes de réutilisation et de centralisation des informations, nous avons implémenté notre GPU, un module unique qui s'occupe de tous les soucis de décodage et d'affichage.
Il est fort polyvalent de sorte à ce que chaque état (FSM) puisse lui transmettre leurs informations et qu'il puisse satisfaire leurs divers besoins d'affichage.

3.3. Implémentation du Keypad (`Keypad_IO`)

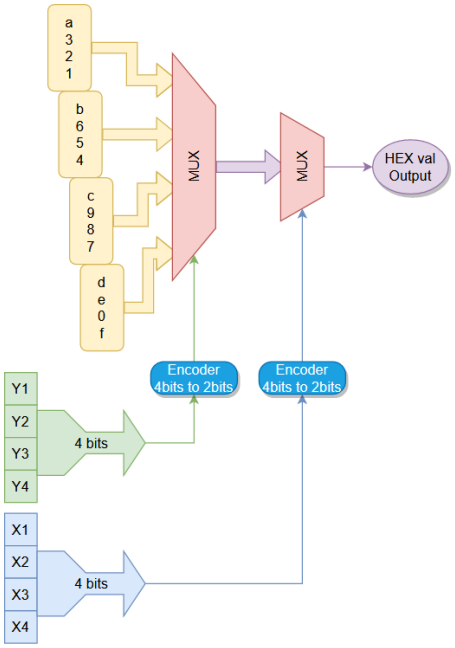
3.3.1. Bubble Counter

Pour implémenter le « Bubble Counter » on a utilisé une FSM qui fonctionne comme « Ring Counter » de sorte a parcourir avec un « 0 » les entrés X's pour trouver la bonne touche pressée sur la matrice du key-pad par « cross matching » des lignes Y's et colonnes X's.



3.3.2. Décodage Affichage Keypad

Afin de décoder l'entrée dune touche du key-pad en valeur digital, on a créé un bloc qui prend les 8 entrées de la matrice du key-pad contenant des multiplexeurs cascader sélectionnant des valeurs fixes. Cela en partant de 4 blocs de 4 valeurs, passant à un bloc (une ligne du key-pad) de 4 valeurs puis a une valeur entre ces 4 (une Touche du key-pad (Y_i,X_i)). On a implémenté un encodeur 4-2bits pour passer la sélection au multiplexeur.

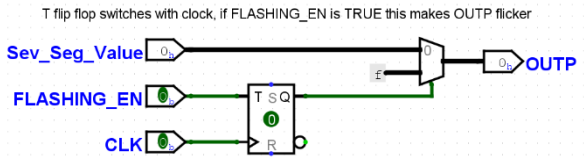


- Lecture et Enregistrement du Keypad (`Set_Reveil` & `Set_Heure`)

Nous avons mis en œuvre une FSM diriger par un conteur suivie dune serie de registre pour garder momentanément les valeurs entrées afin de les enregistrer plus loins dans la mémoire generale du système. Cette FSM est gérée en entrer par un bouton de OK specifique du clavier (#) et par le signal « Data Ready ». (*pas de diagrame)

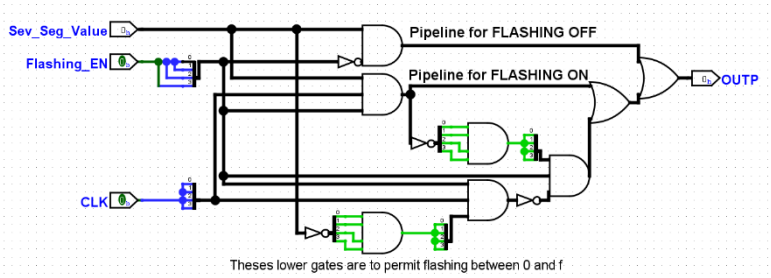
3.4. Module de Clignotage (Flashing_Logic)

Ce module assez bas niveau s'est avéré d'être très intéressant pour notre projet car il était la source d'un **beug majeur de Hardware**.



Logique séquentielle

Le circuit du dessus qui utilise une flip flop T pour alterner la sortie entre l'entrée 7segment encodée sur 4 bits et `0xF` qui était encodé à « tous segments éteints ».



Logique combinatoire

Le circuit du dessous utilise purement une logique combinatoire pour effectuer la même fonction.

Il se trouve que les deux circuits ci-dessus fonctionnent sur le simulateur Logisim-Evolution mais uniquement le circuit combinatoire fonctionne sur la FPGA DE-10 Light !!
Nous avons aussi reconstruit en vain le multiplexeur du circuit séquentiel avec des portes logiques en pensant que c'était le circuit tout fait qui était défectif.
Symptomatiquement, on trouvait nos affichages 7 segments toujours éteint car leurs entrées avant le décodeur étaient toujours à `0xF` peu importe la `CLK`, le `Flashing_EN` ou la valeur de `Sev_Seg_Value`.