Machine de Turing

Rapport de projet électronique

Etudiants: *LAZOUZI Jihane*

ZORZETTO Damien

Encadrant : *BOCQUILLON Ronan*

Année universitaire : *2022/2023*

Sommaire

Introduction générale

a. Contexte du projet

Afin d'avoir un dispositif concret et un outil pédagogique qui simule le fonctionnement de la machine de Turing. Nous essayons à travers notre projet de présenter à l'aide d'une matrice LED, et pour une opération arithmétique donnée, de modéliser le fonctionnement de la machine, en commençant par la saisie des entrées (nombres binaires dans notre cas), et jusqu'aux sorties (l'affichage du résultat de l'opération choisie). La machine doit faire l'opération arithmétique demandée à travers une succession de changement d'états, sans faire l'opération elle-même.

b. Principe de fonctionnement

Etant dans un état initial, la machine décide selon le caractère c pointé par la tête de lecture et l'état actuelle dans lequel elle est :

- Le caractère par lequel le caractère c sera remplacé
- Le déplacement de la tête de lecture
- Le prochain état dans lequel la machine sera

La même procédure s'exécute jusqu'à arriver à un état final, qui lui décide la fin de changement d'états, pour indiquer que le résultat affiché en sortie est bien le résultat attendu.

Pendant l'exécution de changement d'état, la sortie peut changer mais elle ne représente pas le résultat attendu tant qu'elle n'est pas arrivée à un état final, elle est dans des états intermédiaires.

c. Cahier de charges

Une machine capable d'exécuter des opérations arithmétiques sur des nombres binaires (au moins addition dans un premier temps), en suivant le principe de fonctionnement de la machine d'états (machine de Turing) qui doit être visualisé à l'aide d'une matrice LED.

Conception de la machine :

- Des boutons pour :
    1. Un déplacement à droite de la position de la tête de lecture tout au long du ruban (d'une colonne n à une colonne n+1)
    2. Un déplacement à gauche de la position de la tête de lecture tout au long du ruban (d'une colonne n à une colonne n-1)
    3. Un changement de symbole de la case pointée par la tête de lecture au suivant (selon un ordre précis de symboles prédéfinis)
    4. Choix de l'opération arithmétique addition
    5. Choix de l'opération arithmétique soustraction
    6. Choix de l'opération arithmétique multiplication
    7. Choix de l'opération arithmétique division
- Représentation du ruban infini par une matrice LED avec :
    - Une largeur égale à la taille du ruban (supposé fini)
    - Chaque ligne représentera un symbole
    - Une ligne réservée pour la représentation de la position de la tête de lecture

Ainsi le nombre de symboles reconnu par la machine sera égale au nombre de lignes de la matrice LED moins un, et le nombre de symboles significatif en entrée, longueur de la donnée à traiter par la machine, indépendamment de leur signification, données réelles ou contraintes de saisie (séparateurs ..) sera égale au nombre de colonnes.

Machine de Turing

a. Définition formelle

D'abord la machine de Turing comporte :

- Un ruban infini, divisé en cases consécutives et chaque case contient un symbole d'un alphabet donné
- Une tête de lecture/écriture, qui se déplace tout au long du ruban pour lire et écrire les symboles de l'alphabet dans les cases
- Un registre d'état, qui mémorise l'état courant de la machine.
- Une table d'actions, qui indique pour un caractère lu de la case pointée par la tête de lecture et l'état actuelle de la machine :
    - Le caractère à écrire dans cette case
    - Un déplacement vers la gauche ou la droite ou ne pas se déplacer
    - Le nouvel état dans lequel la machine sera

b. Notions et précisions

Alphabet reconnu par la machine revient à dire que si un symbole d'une case de ruban n'appartient à l'alphabet, le symbole ne sera pas reconnu

Le premier état dans lequel la machine se trouve initialement est appelé **état de départ**. Il existe un seul état de départ pour chaque traitement (algorithme)

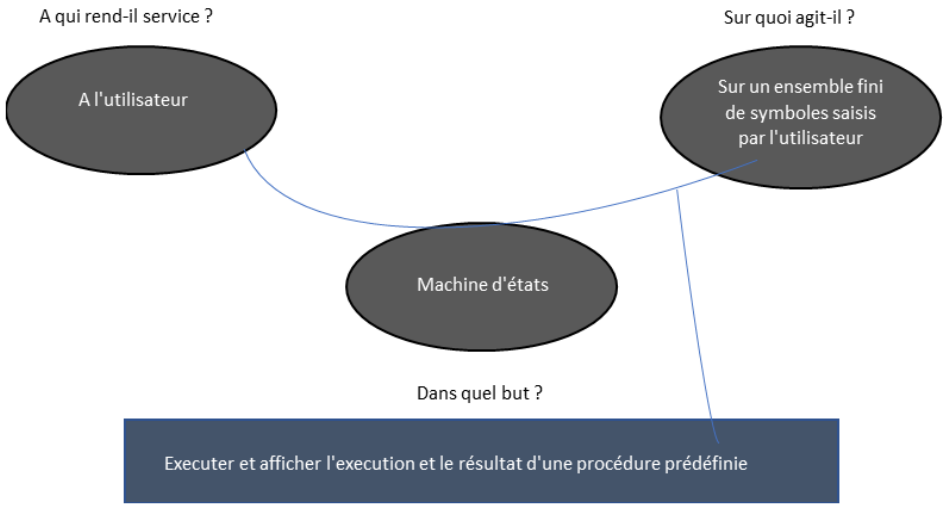
Les **états intermédiaires** sont les états dans laquelle la machine se trouve, avec un changement au niveau de sa sortie, avant que la machine finisse le traitement. Même si les symboles changent en ruban, ils ne reflètent pas le résultat attendu avant qu'un état final soit atteint.

Plusieurs **états finaux** différents en interprétation peuvent exister.

Analyse du projet

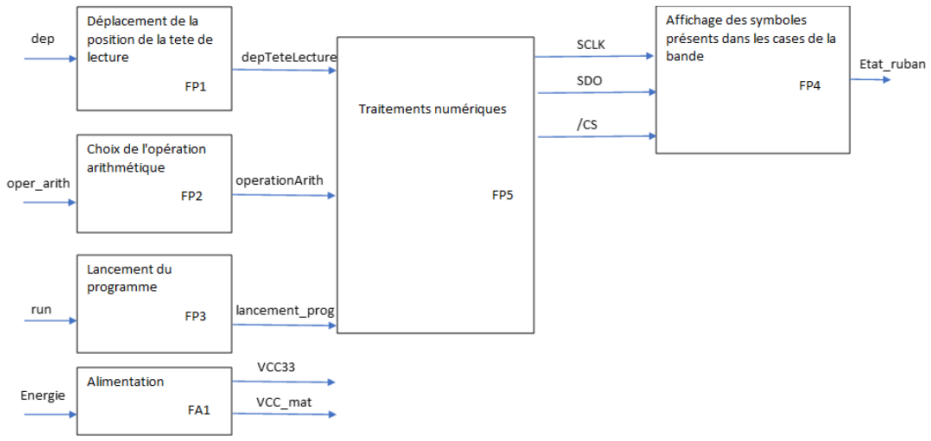
a. Bête à cornes

Pour notre projet, nous avons commencé par une bête à cornes pour préciser à qui notre machine rendra service ? Sur quoi il va agir ? Dans quel but ?



b. Schéma fonctionnel de niveau 1

Pour le schéma fonctionnel de niveau 1, nous avons conçu le schéma suivant :



Cinq fonctions principales ; déplacement de la position de la tête de lecture, choix de l'opération arithmétique, lancement du programme, traitements numériques et affichage des symboles présents dans les cases de la tête de la bande

c. Matrice MOSCOW

Après établir le cahier de charges, le négocier et valider avec le client, nous sommes arrivés aux spécifications :

- Indispensables qui doivent exister dans notre machine (Must Have)
- Obligatoire mais pas indispensables (Should Have)
- Qui peuvent être intégrer dans des prochaines mises à jour (Could Have)
- Qui ne seront jamais intégrer dans notre produit (Won't Have)

Must Have - Machine de Turing capable au moins de faire l'addition de 2 nombres binaires - Possibilité de saisir les entrées (nombres binaires) - Affiche de position de tête de lecture - Déplacement à gauche et à droite de la position de tête de lecture	**Should Have** - Affichage sur 2 matrices LED -Choix entre les 4 opérations arithmétiques - Affichage de changement de symboles et position de tête de lecture lors de l'exécution de la procédure
Could Have - Définition des symboles et états (définissant une procédure) par l'utilisateur	**Won't Have** - Gestion des erreurs de saisie

Conception matérielle

a. Choix de composants

Selon le cahier de charges et des exigences clients, les choix de composants suivants étaient faits :

- **Choix de la matrice**

Pour le choix de la matrice, plusieurs critères ont été pris en compte pour garantir le fonctionnement attendu et satisfaire aux exigences de clients :

- Etant donné que la machine doit faire des opérations arithmétiques sur des nombres binaires, le nombre de symboles pour la machine sera donc 3 (le 0 binaire + le 1 binaire + le caractère nul) + la ligne réservée pour l'affichage de la position de la tête de lecture. Le nombre minimum de lignes de la matrice LED doit être égale à 4.
- Le nombre de colonnes de la matrice représentera la longueur des 2 nombres binaires saisis + 1 colonne pour les séparer (au milieu) + 2 colonnes pour les délimiter de la gauche et de la droite. Les nombres finalement saisis seront sous la forme :

  D_G + X + S + Y + D_D (avec D_G : symbole pour limite gauche, D_D : symbole pour limite droite, S : symbole pour séparer les 2 nombres binaires, X : premier nombre binaire, Y deuxième nombre binaire)

  Ainsi, nous avons choisi 2 matrices LED de taille 8x8 avec un nombre de lignes total égale à 8 et un de colonnes total égale à 16 pour avoir une longueur raisonnable de nombres binaires et pour des LEDs suffisamment grandes pour l'affichage des résultats. Le but étant pédagogique, la taille des LEDs doit être suffisamment grande et un affichage de plusieurs couleurs doit être possible, afin de bien visionner le fonctionnement.

- **Choix du microprocesseur :**

Pour le choix de notre microcontrôleur qui devra contrôler notre matrice un certain nombre de points à guider nos choix :

- Etant donné que le microcontrôleur est censé communiquer avec la matrice il est impératif que ce dernier puisque communiquer en SPI
- Notre machine de Turing comporte un certain nombre de boutons poussoir qui nous permet d'interagir avec la machine, le nombre de boutons poussoir s'élève à un nombre de 7.
- Le microcontrôleur devez avoir une fréquence d'horloge suffisante pour que l'utilisateur les opérations effectuées par le microcontrôleur soit le plus fluide possible
- Le microcontrôleur devait pouvoir accueillir notre programme et avoir un espace mémoire suffisant pour recevoir des modifications futures

À la suite de nos recherches nous avons trouvé un microcontrôleur qui nous permettait de répondre à ces contraintes le PIC24FJ64GA002 :

- ✓ Communication SPI : Le PIC24FJ64GA002 dispose d'un module SPI intégré, ce qui nous permet de communiquer "facilement" avec la matrice via cette interface.
- ✓ Boutons poussoirs : Le PIC24FJ64GA002 dispose d'un nombre suffisant de broches d'entrée/sortie pour accueillir les 8 boutons poussoirs de notre machine de Turing car il peut monter jusqu'à 10.
- ✓ Fréquence d'horloge : Le PIC24FJ64GA002 peut fonctionner jusqu'à une fréquence maximale de 32 MHz, ce qui vous donne une horloge suffisamment rapide pour exécuter nos opérations de manière fluide.
- ✓ Espace mémoire : Le PIC24FJ64GA002 dispose de 64 Ko de mémoire programme (Flash) et de 8 Ko de mémoire RAM, ce qui nous offre un espace suffisant pour stocker notre programme et gérer les données nécessaires à notre application. De plus, il dispose également de 256 octets de mémoire EEPROM pour stocker des données permanentes si nécessaire.

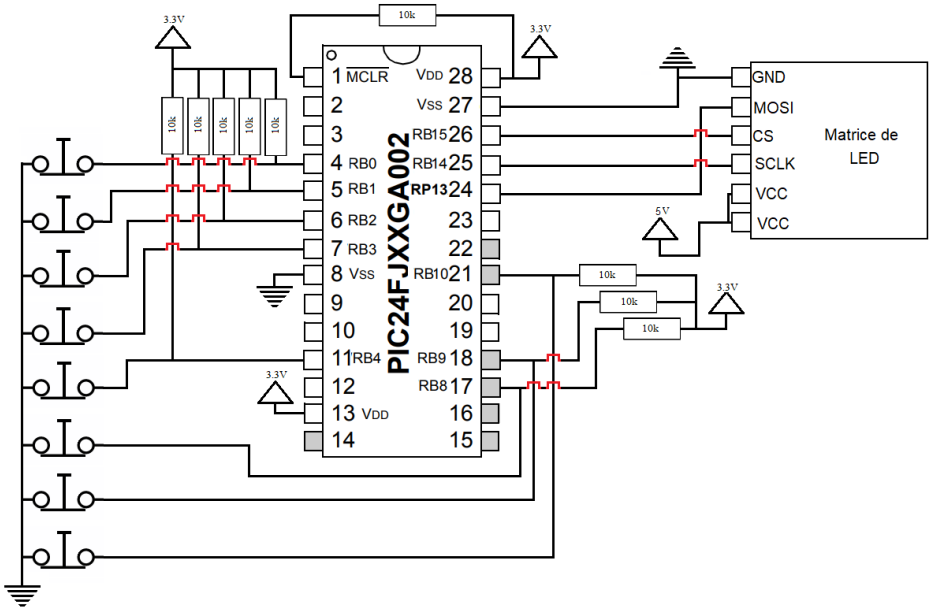
Le **PIC24FJ64GA002** est donc un choix approprié pour notre projet.

b. Assemblage des matrices LED

Les deux matrices LED de 8x8 sont placées côte à côte pour former une matrice plus grande de 8x16. Ces matrices ont une interface SPI, qui est utilisée pour communiquer avec le microcontrôleur. Il est donc nécessaire de les connecter correctement en respectant les instructions du fabricant.

Connexion des boutons poussoirs :
Les 8 boutons poussoirs sont ensuite connectés au microcontrôleur. Ces boutons nous permettront d'interagir avec la machine de Turing, en entrant des valeurs binaires et en contrôlant le déroulement des opérations.

Installation du microcontrôleur :
Le microcontrôleur est l'élément principal de notre système. Il est chargé de contrôler le fonctionnement de la matrice et d'interagir avec les boutons poussoirs. Il est installé sur une carte de prototypage, et les pins correspondants doivent être connectés à la matrice et aux boutons poussoirs. Nous devons respecter les niveaux de tension appropriés pour éviter tout dommage.



c. Communication entre microcontrôleur PIC24 et la matrice LED

La communication entre le microcontrôleur PIC24 et notre matrice de LED dépend du protocole SPI (Serial Peripheral Interface), qui est un protocole de communication série couramment utilisé.

Le protocole SPI utilise quatre fils pour la communication :

- SCLK (Horloge de série) : C'est le signal d'horloge généré par le maître (dans notre cas, le PIC24). Il détermine le rythme de la communication.
- MOSI (Master Out Slave In) : C'est la ligne de données du maître vers l'esclave. Le maître utilise cette ligne pour envoyer des données à l'esclave.

- MISO (Master In Slave Out) : C'est la ligne de données de l'esclave vers le maître. L'esclave utilise cette ligne pour envoyer des données au maître.
- CS (Chip Select): C'est le signal utilisé par le maître pour activer un esclave spécifique. Dans un système avec plusieurs esclaves, chaque esclave aura sa propre ligne CS.

Dans notre projet, nous utilisons le PIC24 en tant que maître SPI. Nous utilisons la broche CS pour signaler à la matrice (l'esclave) le début et la fin de la transmission. Les données sont ensuite envoyées.

La matrice maintient un tampon de 64 octets qui représente chaque position dans la matrice. Pour afficher une image sur la matrice, un ensemble de 64 octets doit être transféré séquentiellement à la matrice en gardant la broche CS à un niveau bas (0V). Chaque octet représente la couleur désirée de chaque LED dans la matrice, avec les 3 premiers bits pour le rouge, les 3 bits suivants pour le vert, et les 2 derniers bits pour le bleu.

Buffer Byte Representing and LED Color Value							
Red			Green			Blue	
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Un ensemble d'exemples de couleur sont fournis dans la documentation technique de la matrice de LED.

Color(Brightest Setting)	Byte Value
Black	0x00
Red	0xE0
Green	0x1C
Blue	0x03
Orange	0xFC
Magenta	0xE3
Teal	0x1F
White	0xFF

Pour que la matrice affiche "noir, rouge, vert, bleu" sur les quatre premières positions de la première ligne, et noir partout ailleurs, voici le processus à suivre:

1) Activer CS (l'amener à un niveau bas)

2) Délai de 0,5 ms
3) Transférer 0x00 via SPI
4) Transférer 0xE0 via SPI
5) Transférer 0x1C via SPI
6) Transférer 0x03 via SPI
7) Transférer 0x00 60 fois via SPI
8) Désactiver CS (l'amener à un niveau élevé)

Conception logicielle

Puisque la machine de Turing est un modèle abstrait de machines mécaniques de calculs tel que l'ordinateur, qui est censé représenter une personne virtuelle exécutant une procédure bien définie, en changeant le contenu des cases d'un ruban infini, en choisissant ce contenu parmi un ensemble fini de symbole, les structures de données de notre programme ainsi que les fonctions exécutées par le microcontrôleur doivent représenter cela.

a. Structures de données

Pour chaque état, définie par son numéro, nous avons :

- Numéro d'état
- Etat final ou intermédiaire
- Pour chaque symbole de l'alphabet, nous attribuons une action définie par :
    - Un nouveau symbole *nv_symbole* par lequel le caractère lu de la case pointée par la tête de lecture sera remplacé
    - Le déplacement de la tête de lecture (gauche, droite ou rien)
    - Un nouvel état de la machine

b. Fonctionnement

Dans un premier lieu, nous nous intéressons juste à l'addition, la succession d'états permettant de faire l'addition de deux nombres binaires est bien définie et disponible sur internet, un exemple d'illustration en annexe 1 la modélise.

Pour l'addition des nombres binaires, nous avons besoin :

- Alphabet composé de 3 symboles et chaque symbole est représenté par une couleur :
    - Le 1 binaire représenté par le rouge

        - Le 0 binaire représenté par le vert
        - Le caractère nul qui nous servira essentiellement pour séparer les 2 nombres binaires représenté par le blanc
    - 7 états tel que le numéro d'un état est son indice dans le tableau et dont le dernier état représente l'état final (numéro d'état allant de 0 à 6)
    - Position de tête de lecture qui sera représentée par le bleu.

Puisque la saisie des 2 nombres binaires par l'utilisateur doit être faire par un déplacement, à gauche ou à droite, tout au long de la matrice, la LED représentant la position de tête de lecture doit se déplacer selon le bouton appuyé par l'utilisateur.

Une fois le bouton de changement de symbole est appuyé, le symbole initialisé au caractère nul représenté par le blanc doit changer au 1 binaire représenté par le vert, puis en 2 binaire représenté par le rouge si l'utilisateur réappuie, et finalement revenir au blanc après un troisième appui.

Finalement, l'utilisateur valide les entrées, lance le programme pour exécuter l'algorithme suivant:

Algorithme d'addition binaire :

Entrée : *bande[M] : tableau de 0, 1 et 2*

Etat: structure composée de :

- numero : entier

- final : booléen

- symbole[N] : tableau de structure Action

Action: structure composée de :

- nouveau_symbole: entier de 0 à 2

- nouveau_etat : entier de 0 à 6

- deplacement : enum {rien, gauche, droite}

Sortie : *bande[M] : tableau de 0, 1 et 2*

Spécifications :

- *M longueur de la bande ;*
- *Pour 2 nombres binaires de taille x et y, bande[M] doit être sous la forme suivante :*
    - *La première et la dernière cases doivent être le symbole nul*

- *Chaque symbole saisi ne doit pas comporter d'autres symboles que le 1 et 2, donc pour le symbole de longueur x, le symbole est exprimé par une suite successive de taille x de 1 et 2.*
- *Les deux nombres binaires sont séparés par le symbole nul*
- *Les cases non utilisées à gauche du premier symbole et à droite du deuxième doivent être remplie par le symbole nul*

Exemple: 00XXX0Y0 avec X= (1 ou 2) et Y= (1 ou 2)

Variables : *i, a : entier*

DEBUT :

Tant que (! Etat[i].final) faire:

a= bande[position_tete_lecture];

bande[position_tete_lecture]= Etat[i].symbole[a].nouveau_symbole:

position_tete_lecture += Etat[i].symbole[a].deplacement;

i= Etat[i].symbole[a].nouveau_etat;

Fin tant que

FIN

Synthèse du projet

a. Organisation du travail

Le diagramme de Gantt suivant représente l'organisation suivant tout au long du projet de la définition des besoins jusqu'à la validation, en suivant le cycle en V.

b. Etat des lieux

A la fin de projet, nous avons une machine de Turing capable de faire l'addition de deux nombres binaires, saisis à l'aide de :

- 1 bouton de déplacement de tête de lecture à gauche
- 1 bouton de déplacement de tête de lecture à gauche
- 1 bouton pour le changement de symbole présent dans la case pointée par la position de tête de lecture

Une fois le programme lancé, une succession de changement d'états est affichée dans la matrice LED, modélisée par un changement de position de tête de lecture et un changement (ou pas) de symbole de la case pointée, jusqu'à arriver à l'état final qui se repère par la stabilité du pointeur de la position de tête de lecture.

Puisque le principe de l'addition consiste à soustraire 1 d'un nombre binaire x et de l'ajouter au nombre binaire y jusqu'à ce que x vaille 0 et y vaille la somme de x et y, le résultat affiché en matrice sera une succession de cases vertes de longueur de x (0 binaire) et une succession de cases vertes ou rouges ou les deux selon le résultat (0 et 1 binaire) séparées par une case blanche (symbole nul).

Conclusion

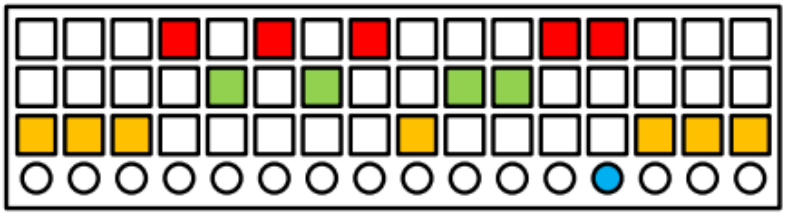
Afin de satisfaire au besoin de client, il fallait d'une part garantir que la machine aura exactement le fonctionnement d'une machine d'états (conception logicielle), et d'autre part garantir la compatibilité des composants en leurs interactions et en réponse aux exigences du client (conception matérielle).

En faisant certains choix dirigés par quelques exigences et contraintes, de nouvelle fonctionnalités et possibilités d'application ont été offertes, par exemple en le choix de la matrice LED permet d'avoir un nombre plus grand de symboles et fonctions.

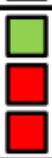
Annexe 1 :

Titre : Addition de 2 nombres binaires

Etat initial de l'afficheur



Programme de l'algorithme

Etat Actuel		Etat Futur		
N° d'état	Couleur	Couleur	Flèche	N° d'état
1				2 1 1
2				2 2 3
3				3 4 4
4				4 4 5
5				6 5 5
6				6 6 1