Conception et réalisation d'un processeur 8 bits

Nathan Soufflet

17 mars 2018

1 Introduction

Ce projet consiste en la conception puis réalisation d'un ordinateur Turing-complet à partir de composants discrets élémentaires ainsi que l'ensemble des composantes logicielles permettant son utilisation pratique. L'objectif principal est la visualisation et compréhension profonde des différentes opérations et couches impliquées dans l'éxécution d'un programme ecrit dans un langage haut niveau.

2 Partie matérielle

2.1 Organisation en modules

Le processeur suit une architecture de von Neumann avec un bus de données 8 bits et un bus d'adresse 16 bits. Il est structuré en 9 modules distincts réalisant chacun une fonction très simple contrôlable à l'aide de microcodes, cette approche possède de nombreux avantages :

— Segmentation du travail à fournir
— Possibilité de tester chaque fonction séparément
— Possibilité de faire évoluer chaque module individuellement
— Feedback rapide durant la conception

Il se pose toutefois un problème : Comment connecter chaque module aux bus et comment faire en sorte que le module de contrôle ait accès à tous les modules ?

En effet, du fait de la géometrie rectangulaire des modules, chacun d'entre eux ne peut accéder au maximum qu'aux quatre modules connexes. Une solution serait alors de faire passer des connexions en-dessous ou au-dessus de certains d'entre-eux avec des câbles. Ce n'est cependant pas très élégant et peut entraîner des erreurs de branchement. Une autre solution a été choisie : les modules se chargeront, en plus de remplir leur fonction, de propager des signaux aux modules connexes.)

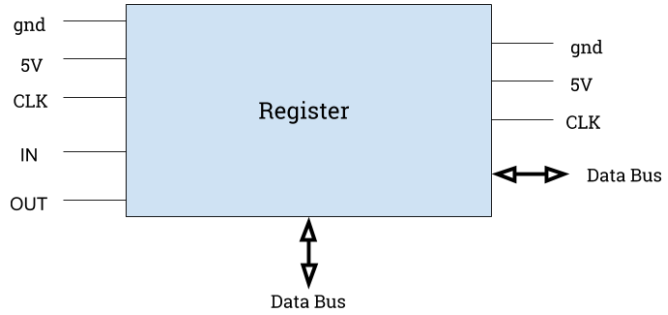

Figure 1 – Propagation du signal d'horloge et du bus de données d'un module de registre

Une étude sera alors menée sur la disposition optimale des modules permettant de minimiser le nombre de signaux à propager ainsi que leur distance.

2.2 Architecture

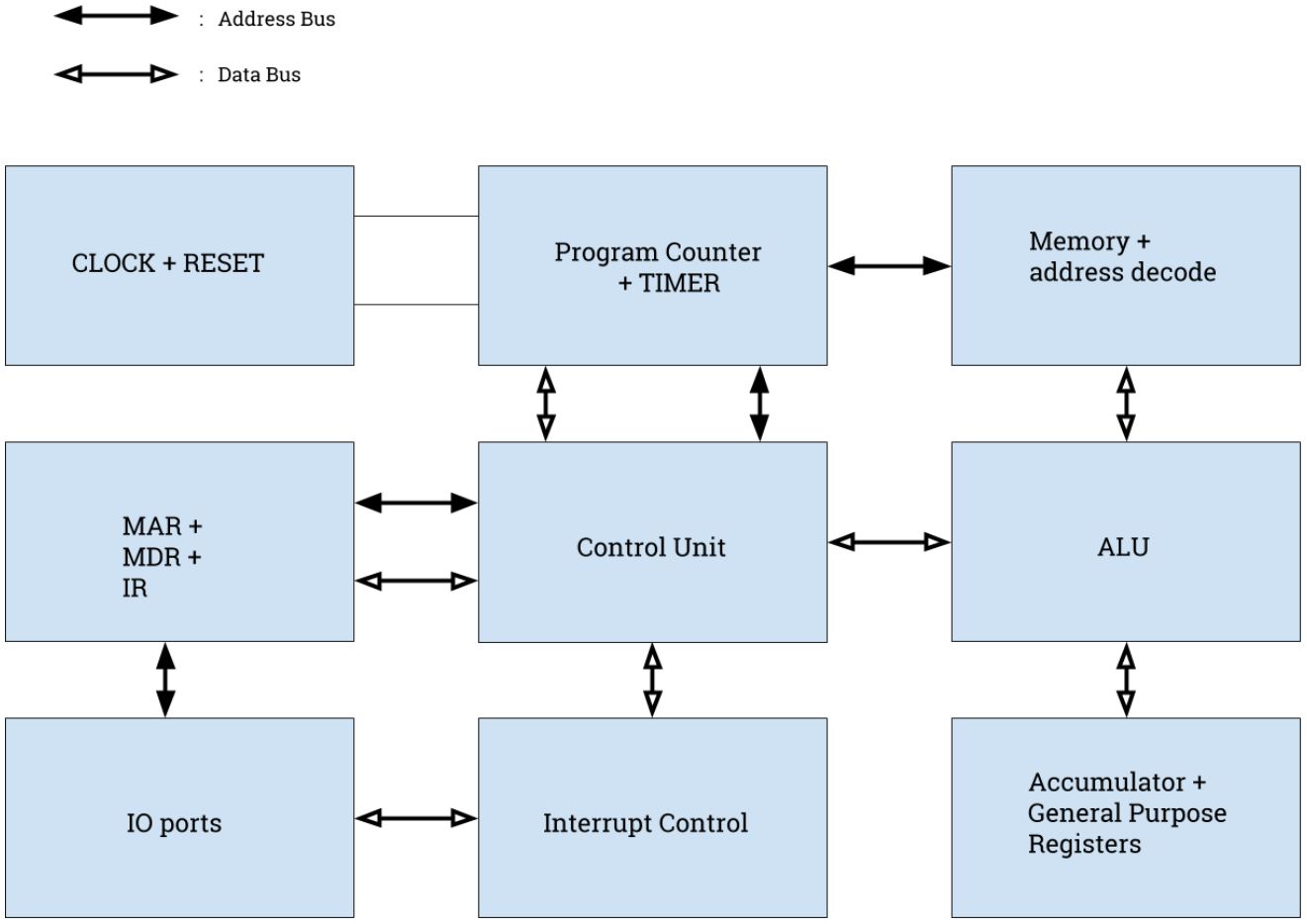

FIGURE 2 – Organisation possible des modules

2.2.1 CLOCK + RESET

Ce module s'occupe de la génération d'un signal carré périodique permettant de synchroniser les opérations des différents modules entre-eux. Il dispose de trois modes d'opération :

- Lent : Permet de faire varier la fréquence entre 0.8 et 400Hz
- Rapide : Permet de faire varier la fréquence entre 1kHz et 1.6MHz
- Manuel : Permet de contrôler manuellement l'horloge (mode pas à pas)

La seconde fonction de ce module est de réinitialiser le processeur pendant la première demi-seconde après allumage (de sorte à ce que la tension se stabilise à 5V garantissant le bon fonctionnement des modules) et à chaque appui d'un bouton dédié.

2.2.2 Program Counter + Timer

Ce module contient deux registres 16 bits :
- Program Counter : Contient l'adresse de la prochaine instruction à décoder
- Timer : Contient le nombre de cycles écoulés depuis l'allumage (modulo 2^{16})

2.2.3 Memory + Address decode

Ce module contient la ROM (32k premiers octets : mémoire contenant les instructions) et la RAM (32k derniers octets : mémoire contenant les données) ainsi que la logique de décodage d'adresse permettant de selectionner la banque mémoire dans laquelle lire ou écrire des données.

2.2.4 MAR + MDR + IR

Ce module contient trois registres :
- Instruction Register (IR - 8 bits) : Contient la prochaine instruction à exécuter
- Memory Address Register (MAR - 16 bits) : Contient l'adresse selectionnée dans le module mémoire
- Memory Data Register (MAR - 16 bits) : Contient un octet de données à écrire dans la mémoire

2.2.5 Control Unit

Ce module décode l'instruction contenue dans le registre d'instruction et active les bits de contrôle correspondants.

2.2.6 ALU - Arithmetic-Logic Unit

Ce module peut effectuer 16 opérations arithmetiques ou logiques sur l'accumulateur et / ou un autre registre 8 bits.

2.2.7 IO ports

Ce module permet d'échanger des données avec l'extérieur comme un clavier ou un écran.

2.2.8 Interrupt control

Ce module s'occupe de la gestion des intérruptions, il contient :
- Interrupt Sub Routine Register (ISR - 16 bits) : Contient l'adresse de la première instruction de la routine de gestion des interruptions
- Interrupt Code Register (ICR - 8 bits) : Contient le code de l'interruption

Un tel module permet de gérer efficacement les evenements d'entrées sorties, par exemple, dès qu'une touche d'un clavier est pressée, un signal d'interruption est levé, à la fin de l'execution de l'instruction en cours, ce module enregiste l'état des registres les plus importants sur le stack et initialise le Program Counter à l'adresse contenue dans l'ISR.

La routine de gestion des interruptions se charge alors de traiter l'interruption en fonction du code contenu dans l'ICR.

2.2.9 Accumulator + General Purpose Registers

Ce module contient 8 registres 8 bits :
- Accumulator (A) : Contient le premier opérande de l'ALU
- B Register (B) : Contient le deuxième opérande de l'ALU pour certaines opérations
- C Register (C) : Registre à utilisation générale
- D Register (D) : Registre à utilisation générale
- X Register (X) : Registre à utilisation générale
- Y Register (Y) : Registre à utilisation générale
- Flags Register (F) : Registre contenant les différents bits d'état (Carry, Zero, Overflow...)
- Stack Pointer (S) : Registre pointant vers la prochaine adresse du stack

2.3 Choix des composants

Il est en théorie possible de réaliser ce projet à partir de circuits très simples à base de transistors discrets, de résistances et de condensateurs uniquement, cependant en pratique de tels circuits disposent de nombreux problèmes, entre autres : aucune protection de surtension/surintensité, perte de courant en chaînant plusieurs circuits pouvant entraîner la non-transition de transistors à l'état haut et bien d'autres. De plus, les transitors

discrets ne sont pas très efficaces en énergie et ne peuvent pas changer d'état à plus de quelques centaines de kilohertz. Nous allons donc préférer l'utilisation de circuits intégrés implémentant des fonctions logiques très simples et s'occupant des détails d'ordre électronique.

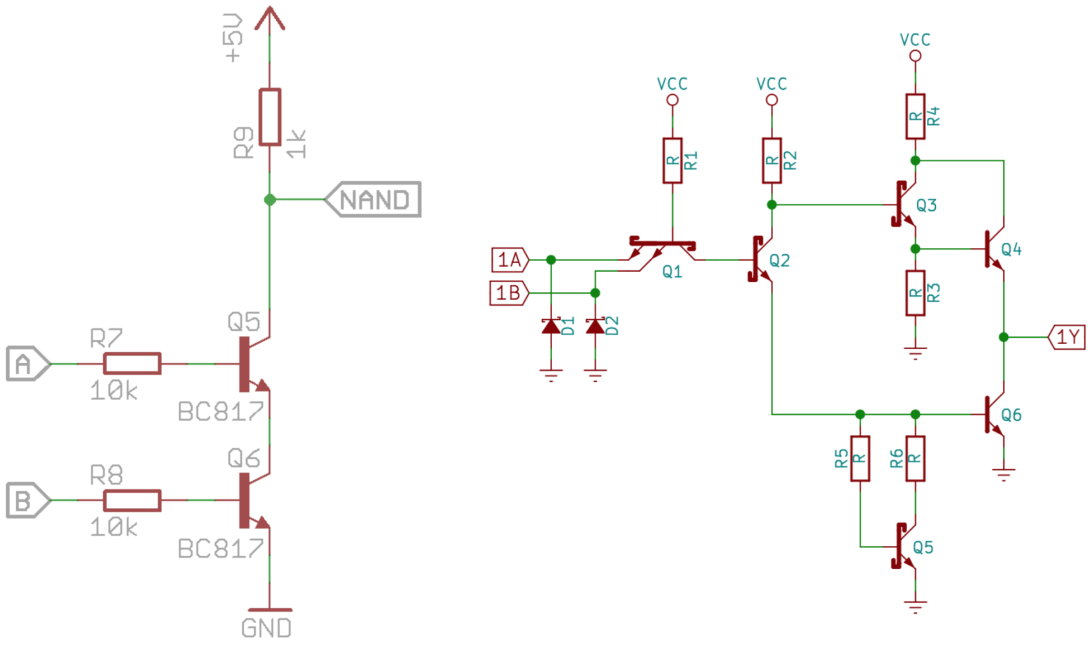

FIGURE 3 – Implémentation naïve vs réèlle (74LS00) d'une porte NAND

Ainsi, seuls les diagrammes logiques de ces circuits intégrés seront considérés :

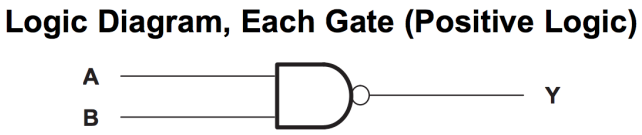

FIGURE 4 – Diagramme logique d'une NAND

Ce niveau d'abstraction permet de se focaliser sur le fonctionnement logique du processeur. La série de circuits intégrés 7400 de Texas Instruments sera utilisée car ceux-ci sont facilement trouvables, peu-chers et étaient très populaires durant l'age d'or des micro-ordinateurs. Plus précisément, nous utiliserons la sous-famille 74HCXX qui dispose d'une faible consommation énergétique grâce à la technologie CMOS et permet d'opérer à plusieurs MHz. Il faut toutefois faire attention à ne pas utiliser de composants de la sous-famille 74LSXX, très répendus mais qui utilisent des niveaux logiques incompatibles.

2.4 Phases de conception

Le bon fonctionnement global du processeur ne pouvant être verifié qu'une fois les 9 modules achevés, il est préférable de simuler chaque module informatiquement dès le début de sorte à s'assurer qu'aucun problème d'ordre logique n'est présent dans l'implémentation proposée.

2.4.1 Implémentation logique

Le logiciel de simulation de circuits logiques Logisim permet d'encapsuler un ensemble de portes logiques en un unique composant permettant ainsi d'integrer aisément plusieurs modules de façon lisible.

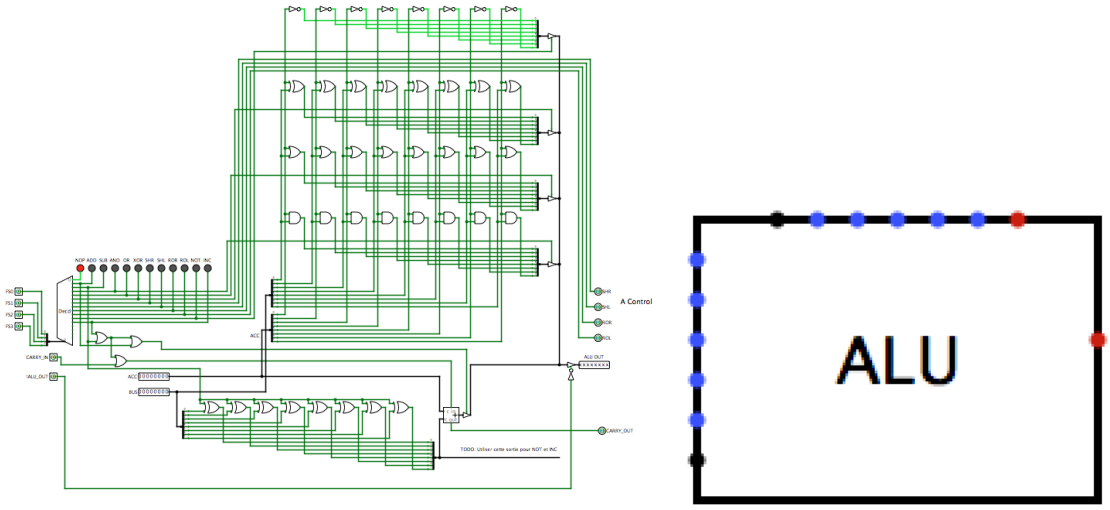

FIGURE 5 – Module ALU simulé à l'aide de Logisim

2.4.2 Première implémentation physique

L'implémentation physique d'un module est plus complexe que la modelisation fonctionnelle, elle contient notamment davantage d'éléments (condensateurs, resistances, boutons...). De plus, les fonctions logiques étant implémentées à l'aide de circuits intégrés, il est nécessaire de s'assurer qu'ils soient branchés et utilisés correctement. Pour cela, une première implémentation physique est réalisée sur des breadboard.

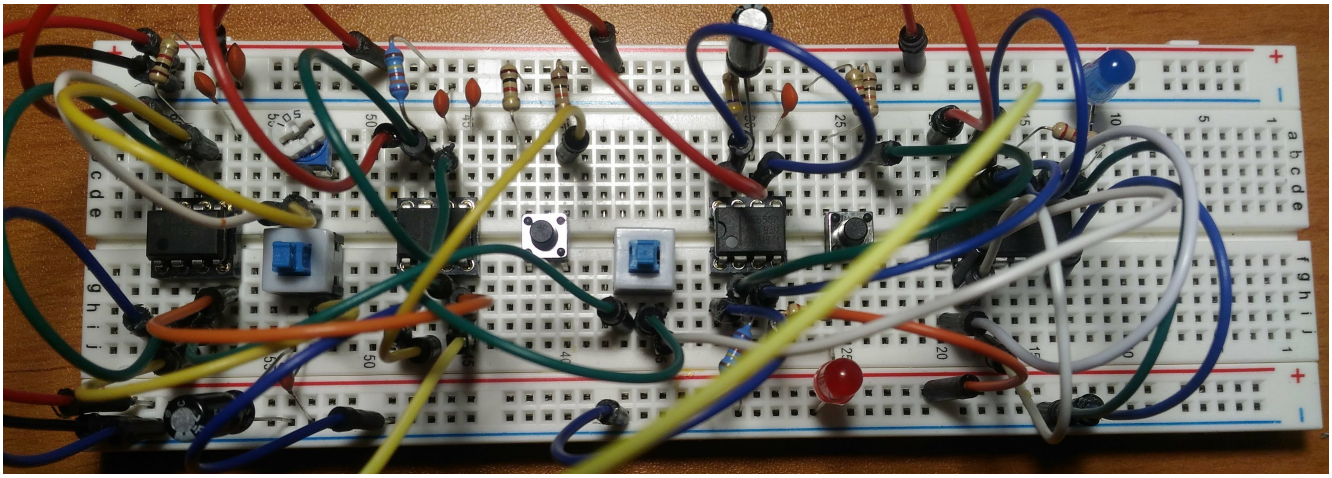

FIGURE 6 – Module d'horloge et de reset implémenté sur une breadboard

2.4.3 Conception du PCB

L'implémentation sur breadboard sert enfin comme référence pour les branchements lors de la réalisation des circuits sur Eagle. Une fois le routage effectué, un fichier gerberx est généré pour être lu par la graveuse à circuits de l'UTT, les composants et connecteurs sont ensuite soudés, des sockets DIP sont utilisés pour les différents circuits intégrés afin de permettre leur utilisation sur une version ultérieure ou pour les changer en cas de non-fonctionnement.

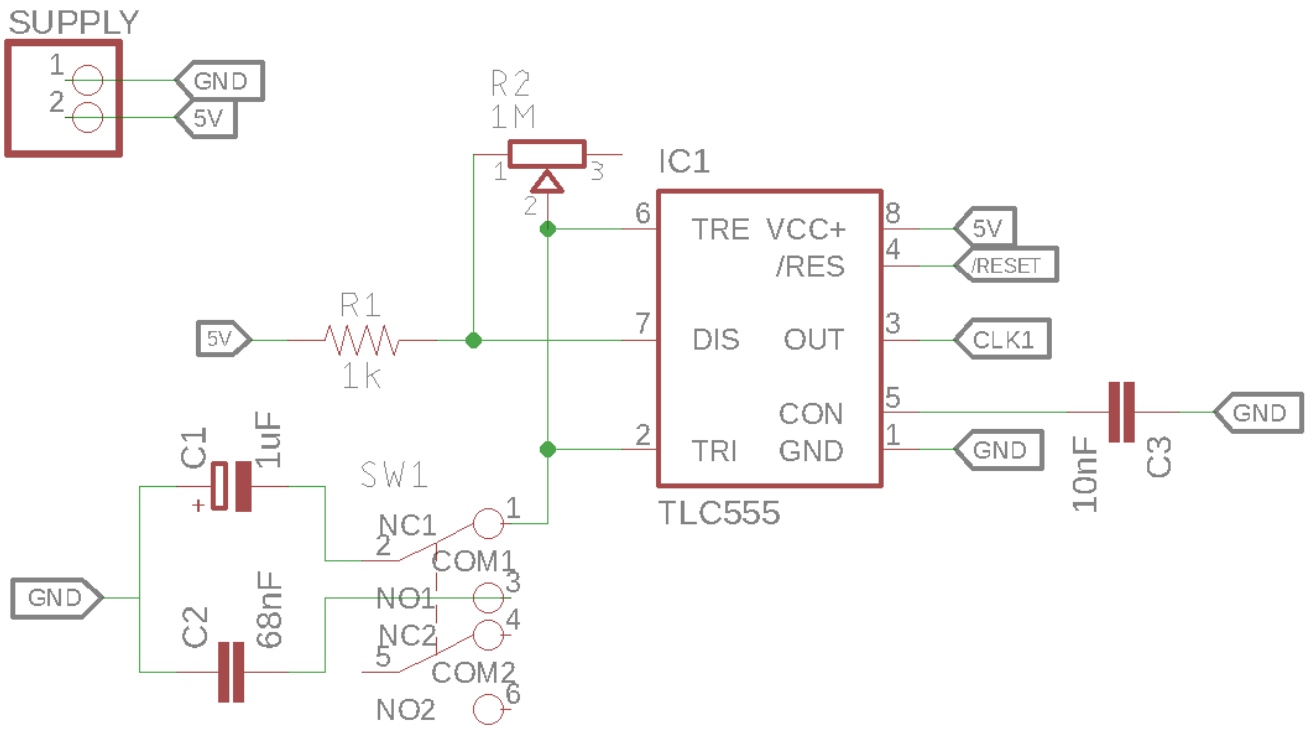

FIGURE 7 – Circuit de génération d'un signal carré de fréquence variable pour le module d'horloge

3 Partie logicielle

La seconde partie du projet consiste à batir progressivement l'ensemble des programmes permettant une utilisation pratique de l'ordinateur, c'est à dire :

- Un système d'exploitation basique s'occupant d'initialiser les registres, ainsi que de gérer les interruptions
- Un assembleur réalisant la traduction d'instructions textuelles en binaire
- Un compilateur d'un subset restreint du langage C