SETR-E2 Conception de circuits

Document de conception contrôleur d'interruptions

Louison Gouy et Mathis Briard
November 15, 2022

École Polytech de Nantes
Electronique et Technlogie Numérique

Enseignant référent : Sébastien LE NOURS

Abstract

Abstract

Contents

1 Introduction **5**
1.1 Contextualisation 5

2 Cahier des charges **6**
2.1 Objectif du circuit 6
2.2 Fonctionnalités attendues 6
2.3 Utilisation du circuit 6
2.4 Chronogrammes caractéristiques 9
2.5 Contraintes du projet 11

3 Spécifications **13**
3.1 Caractérisation de l'environnement 13
3.2 Entrées et sorties du composant 14
3.3 Spécifications fonctionnelles 16
3.4 Spécification des registres 17
3.5 Écriture des procédures de base pour l'emploi du circuit 20
3.6 Spécifications opératoires 20
3.7 Spécifications technologiques 20

4 Conclusion **21**

Acronyms **22**

5 Appendix **24**

List of Figures

1 Schéma de câblage de l'IP à concevoir, du contrôleur mémoire et du processeur 6
2 Entrées et sorties de l'IP à concevoir 7
3 Chronogramme spécification du bus 9
4 Chronogramme spécification signal IT au processeur 10
5 Chronogramme spécification 2 signaux IT cas 1 10
6 Chronogramme spécification 2 signaux IT cas 2 11
7 Planification du projet avec ses jalons 11
8 Comportement du système à microprocesseur 13
9 Comportement de l'entité source d'interruptions 14
10 Entrées et sorties du circuit à concevoir 14
11 Automate du contrôleur d'interruptions 17

List of Tables

1 Sens et rôle des signaux 8
2 Sens et rôle des signaux 14
3 Sous-relations et type de données 15
4 Synthèse des spécifications fonctionnelles 16
5 Vecteurs d'interruptions 17
6 Organisation des registres du contrôleur d'interruption (vue globale) 18

7 Détails du registre CTRL_IT_EN 19
8 Détails du registre CTRL_IT_PEND 19
9 Détails du registre CTRL_IT_BRA 19
10 Détails du registre CTRL_IT_ADDR_N 19
11 Détails du registre CTRL_IT_PRIO_N_N+1 19

Résumé

La méthodologie de conception de systèmes électroniques est une discipline essentielle lorsqu'il s'agit, dans le cadre d'une future carrière d'ingénieur, de réaliser un produit répondant aux besoins d'un client. Et ce domaine est d'autant plus indispensable dans un monde où les systèmes numériques sont de plus en plus complexes et leur conceptions demandent une bonne structuration et une réelle méthodologie. Ce rapport s'inscrit dans la présentation d'une conception d'un controlleur d'interruption par l'utilisation d'une méthode précise, applicable dans un ensemble de secteurs d'activités comme l'automobile, le médical, l'aéronautique, maritime, que ce soit dans le civil ou le militaire.

1 Introduction

1.1 Contextualisation

La formation ETN (Électronique et technologies numériques) offerte par l'école polytechnique de l'Université de Nantes propose d'aborder diverses branches de l'électronique, du traitement du signal au systèmes à microprocesseur en passant par l'électronique analogique des hautes-fréquences. Cet ensemble de domaines techniques nécessite des compétences en matière de méthodologie de conception. Ce rapport s'inscrit dans la conception d'un appareil de marquage routier avec la méthode MCSE. La méthode MCSE (Méthode de conception des systèmes électroniques), née à Ireste par l'impulsion de Jean-Paul Calvez, cette méthode a été implantée au sein d'un outil nommée CoFluent rachetée par Intel® depuis 2011. Cette méthode fait désormais partie de la culture de la formation et constitue l'outil de conception premier de l'ingénieur ETN.
Ce rapport se décompose en diverses parties. Il s'agira dans un premier temps de rappeler le cahier des charges de la conception ..... Pour l'ensemble de ce rapport, les diverses phases de spécifications et conceptions s'appuient sur les deux ouvrages de Jean-Paul Calvez. [**?**] [**?**]
Ce travail de conception à pour but de placer les étudiants dans un contexte industriel. Le cahier des charges fourni prend la forme d'un exemple réel où les spécifications du client sont exprimées.

2 Cahier des charges

Cette partie présente le cahier des charges du périphérique. Elle intègre les quelques points fournis par le sujet auquel s'ajoutent les contraintes imaginées par les étudiants.

2.1 Objectif du circuit

Le contrôleur d'interruptions a pour rôle d'informer le processeur sur l'occurrence d'une interruption valide. Il fournira alors l'adresse de la prochaine instruction à exécuter.

2.2 Fonctionnalités attendues

Les fonctions de service du circuit sont :

1. Masquer et démasquer chaque interruption individuellement
2. Contenir et permettre la configuration du vecteur d'exception
3. Permettre de configurer la priorité des interruptions
4. Ne fournir au Central Processing Unit (CPU) que les interruptions valides
5. Prendre en compte les priorités dans la génération des demandes au CPU

2.3 Utilisation du circuit

Il s'agit ici de donner l'utilisation du circuit en présentant le schéma de câblage du contrôleur d'interruptions ainsi que la définition des signaux logiques d'entrées et sorties. L'Intelectual Property (IP) à concevoir s'interface avec le bus de données, le bus d'adresses ainsi qu'un ensemble de signaux de temporisation et de contrôle. Il est possible de retrouver des signaux de temporisation comme le nWAIT et des signaux de contrôle comme le nRST ou le RnW. Leur rôle est présenté table (**??**).

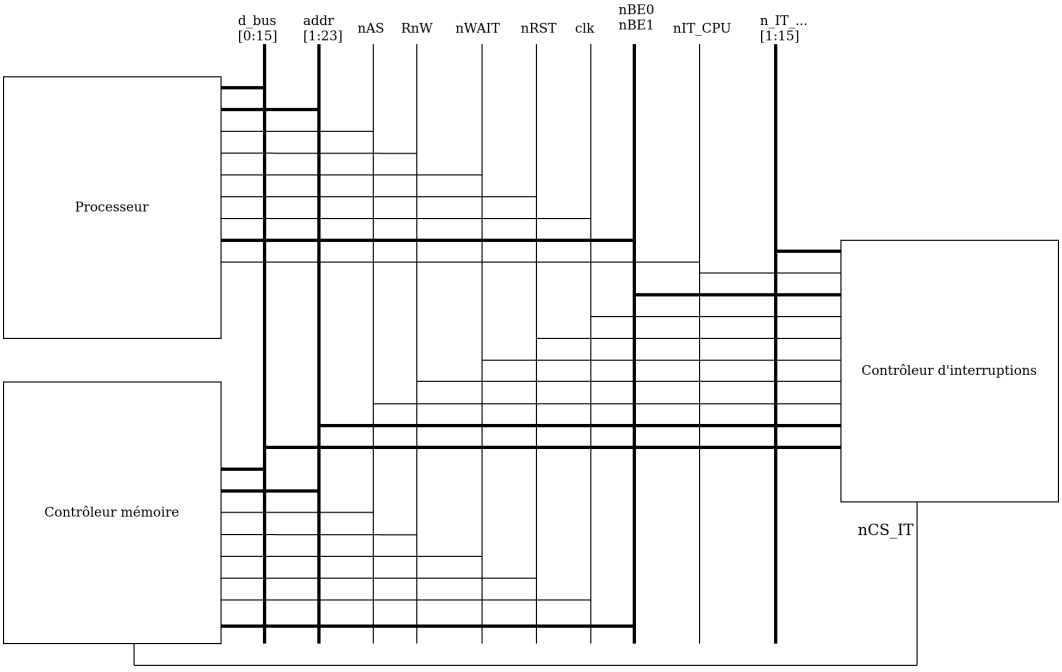

Figure 1: Schéma de câblage de l'IP à concevoir, du contrôleur mémoire et du processeur

Le contrôleur d'interruptions est également câblé avec plusieurs autres IPs du SoC. Il y a par exemple le processeur avec lequel le signal nIT_CPU est commun. D'autre part le contrôleur mémoire est connecté avec le contrôleur d'interruptions par le signal nCS_IT. L'ensemble des périphériques du SoC peut également envoyer un signal d'interruption représenté par nIT_ . . .

Le schéma présenté ci-dessus ne précise pas le sens des signaux. Il n'est donc pas possible de savoir quelles sont les entrées et sorties du contrôleur d'interruptions. Également, les noms des 4 interruptions externes et des 11 autres provenant de divers périphériques ne sont pas donnés. La figure **??** et le tableau **??** présentent les entrées et sorties du point de vue de l'IP à concevoir ainsi que le rôle de chaque signaux.

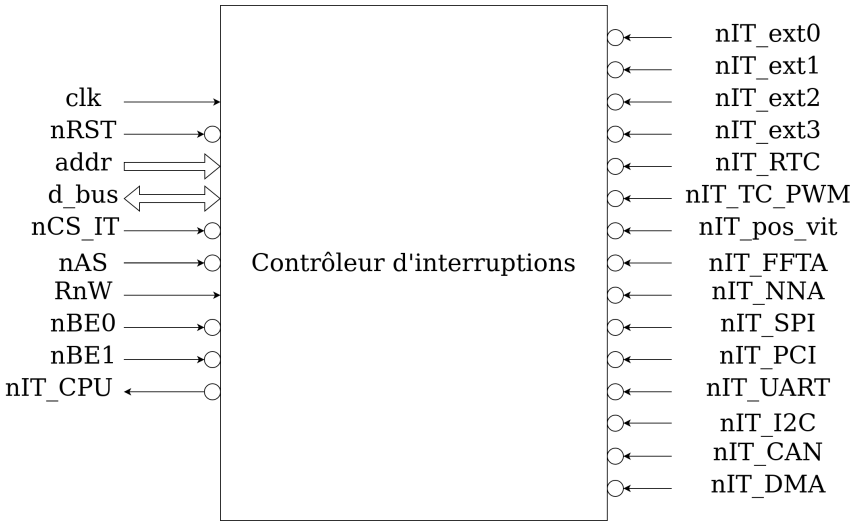

Figure 2: Entrées et sorties de l'IP à concevoir

Nom	Sens	Rôle
clk	Entrée	Signal d'horloge
nRST	Entrée	Signal de réinitialisation
addr	Entrée et sortie	Bus d'adresses
d_bus	Entrée et sortie	Bus de données
nCS_IT	Entrée	Signal de sélection du périphérique en cas d'opérations de lecture ou d'écriture
nAS	Entrée	Signal indiquant la présence d'une valeur sur le bus d'adresse
RnW	Entrée	Signal d'écriture ou de lecture 0 : écriture 1 : lecture
nBE0	Entrée	Signal indiquant la structure de la mémoire 0 : little-endian 1 : big-endian
nBE1	Entrée	Signal indiquant la taille de la donnée 0 : 8 bits 1 : 16 bits
nIT_CPU	Sortie	Signal pour le processeur avertissant qu'une interruption est demandée de la part d'un périphérique
nIT_ext0	Entrée	Signal d'interruption extérieure numéro 0
nIT_ext1	Entrée	Signal d'interruption extérieure numéro 1
nIT_ext2	Entrée	Signal d'interruption extérieure numéro 2
nIT_ext3	Entrée	Signal d'interruption extérieure numéro 3
nIT_RTC	Entrée	Signal d'interruption provenant du périphérique RTC
nIT_TC_PWM	Entrée	Signal d'interruption provenant du Timer et PWM
nIT_pos_vit	Entrée	Signal d'interruption provenant du périphérique de mesure position et vitesse
nIT_FFTA	Entrée	Signal d'interruption provenant de l'accélérateur transformée de Fourier discrète
nIT_NNA	Entrée	Signal d'interruption provenant de l'accélérateur réseau de neurones
nIT_SPI	Entrée	Signal d'interruption provenant du périphérique de communication SPI
nIT_PCI	Entrée	Signal d'interruption provenant du périphérique de communication PCI
nIT_UART	Entrée	Signal d'interruption provenant du périphérique de communication UART
nIT_I2C	Entrée	Signal d'interruption provenant du périphérique de communication I2C
nIT_CAN	Entrée	Signal d'interruption provenant du périphérique de communication CAN
nIT_DMA	Entrée	Signal d'interruption provenant du périphérique d'accès direct à la mémoire.

Table 1: Sens et rôle des signaux

Les noms des signaux présentent un suffixe n signifiant que ceux-ci sont actifs à l'état bas. Par exemple, le signal de sélection nCS_IT est actif à l'état bas. Ainsi un signal de sélection à l'état logique 0 signifie que le contrôleur d'interruptions est sélectionné pour une opération de lecture ou d'écriture dans un des registres.

Conventionner ces signaux comme actif à l'état bas n'est pas anodin. Historiquement,

pour des anciennes technologies Transistor Transistor Logic (TTL), les signaux actifs à l'état bas sont davantage robustes aux bruits. À titre d'exemple, un signal de sélection CS actif à l'état haut sélectionne un périphérique lorsque celui-ci est à l'état logique 1. Pour des raisons purement physiques (glitch sur le signal, baisse de tension à cause de la résistivité des interconnexions, chute de l'alimentation à cause d'un fort tirage de courant), ce signal à l'état 1 peut passer à l'état de haute impédance Z, voire à l'état bas. Un tel problème causerait la perdre de données à lire où écrire mais plus généralement des problèmes de sécurité.

2.4 Chronogrammes caractéristiques

Cette partie détaille les contraintes temporelles via à vis des entrées sorties. Il est possible d'identifier deux types d'échanges pour ce périphérique. Le premier via le bus est illustré avec le chronogramme figure **??**.

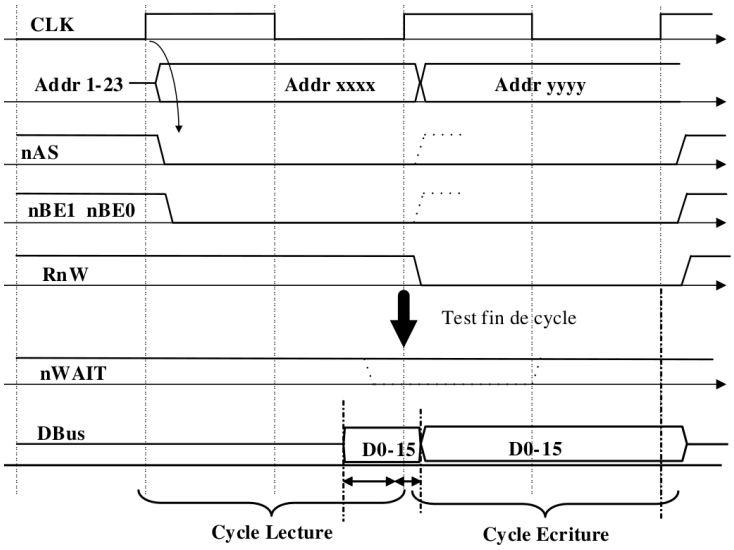

Figure 3: Chronogramme spécification du bus

On retrouve une partie des signaux définis dans la figure **??**. Lors du premier front d'horloge une valeur est placée sur le bus d'adresse comme l'indique sa valeur et nAS. Le périphérique doit alors lire ou écrire la valeur correspondante sur le bus de données avant le prochain front montant. C'est ce qu'indique la flèche centrale sur cette figure. Respecter cette contrainte permettra d'assurer la compatibilité entre les entités communiquant sur le bus. Ce dernier permet un échange bidirectionnel entre le processeur et le contrôleur. Il servira à accéder par lecture et écriture aux différents registres. Parmi eux, on identifie le vecteur d'exception qui sauvegarde les adresses mémoire auxquelles le processeur va brancher. Le registre de masquage permettra de désactiver individuellement chaque entrée. La priorité par défaut de chaque entrée pourra également être modifiée.

Le second type de communication a pour but de notifier le CPU lors d'interruptions. Elles peuvent provenir des autres périphériques présents dans le microcontrôleur ou de sources externes. L'objectif est d'informer le processeur via le signal nIT_CPU en suivant le régime de priorités. La politique de sensibilité du périphérique est l'état des signaux et non les fronts. La gestion d'une interruption commence lorsque le signal émmeteur passe à l'état bas et s'achève lorsque qu'il passe à l'état haut. L'état du signal doit être maintenu au minimum pendant un cyle d'horloge pour qu'il soit valide.

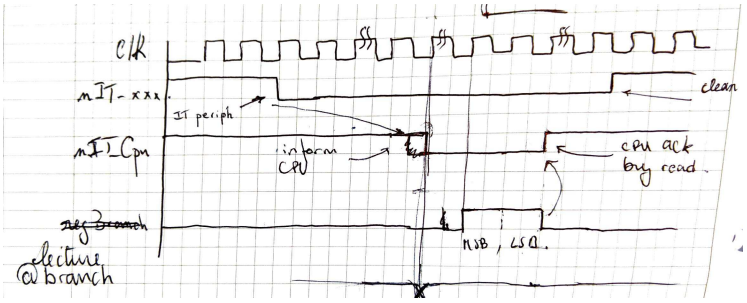

Figure 4: Chronogramme spécification signal IT au processeur

Le chronogramme figure **??** illustre le cas d'un signal d'interruption provenant d'un périphérique et la séquence induite concernant le signal nIT_CPU. Une demande d'interruption s'identifie par le passage à l'état bas de nIT_xxx. Après quelques cycles de traitement visant à confirmer la validité de la requête en consultant le registre de masquage, le processeur est informé, lui aussi, par le passage à l'état bas sur nIT_CPU. Il sauvegarde alors le contexte et lit le registre de branchement. Cette action agit comme un **acquittement** et le signal nIT_CPU est passé à l'état haut. Le processeur se charge d'informer le périphérique émetteur, souvent à la fin de la routine, afin qu'il baisse le drapeau dans son registre d'état. Tant que cela n'est pas fait, aucune demande d'interruption de niveau inférieur ou égal ne sera transmis. Le chronogramme figure **??** illustre justement le cas d'une demande **moins prioritaire** qui apparaît durant l'exécution d'une première.

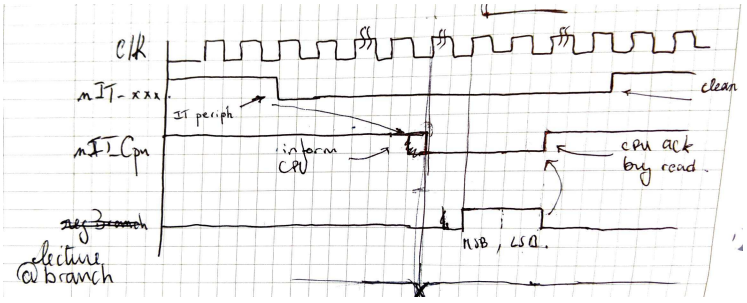

Figure 5: Chronogramme spécification 2 signaux IT cas 1

Dans cette situation, le périphérique d'indice 1 est plus prioritaire que celui d'indice 2. La demande est faite comme précédemment en passant le signal nIT_xx1 à l'état bas. Le contrôleur réagit alors en informant le processeur. Pendant cette opération, une seconde interruption de niveau inférieur apparaît. Aucune réaction n'est attendu de la part du contrôleur qui attend la fin de l'exécution précédente pour générer un front descendant sur nIT_CPU. Le traitement se déroule alors comme auparavant. Il est intéressant de porter l'attention sur la place que joue le signal nIT_xx1 dans ce cas. Son passage à l'état haut autorise de nouveau l'occurrence des interruptions de niveau inférieur ou égal. Il reviendra donc au développeur de placer cet acquittement en fin de routine d'interruption sans quoi la priorisation ne sera pas garantie. Pour poursuivre la description des contraintes temporelles, on peut identifier un second cas venant de paire avec le précédent. Le chronogramme figure **??** illustre la situation où une demande **plus prioritaire** apparaît durant l'exécution d'une première.

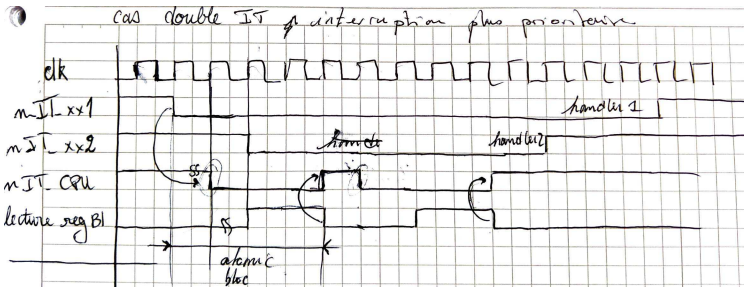

Figure 6: Chronogramme spécification 2 signaux IT cas 2

De nouveau, dans ce cas, le périphérique d'indice 1 est plus prioritaire que celui d'indice 2. Une première demande provient du signal nIT_xx2. Mais durant son traitement, il apparaît, une seconde interruption de niveau supérieur. Le contrôleur attend alors la lecture du processeur pour la première avant de l'informer pour la seconde. Cela est primordial pour éviter toute incohérence dans l'état des registres. Lorsque le processeur a effectué la lecture servant d'acquittement, il est de nouveau notifié. Le signal nIT_CPU doit alors rester au minimum un cycle horloge à l'état bas afin de générer un front descendant synchrone. La suite du cycle se passe comme précédemment pour le contrôleur d'IT. C'est le processeur qui gère le rétablissement du contexte de l'interruption de plus faible priorité.

Nous venons de présenter trois situations de gestion d'interruptions. Cette analyse est loin d'être exhaustive, principalement en raison de l'infinité des instants d'occurrence possibles. Il semble toutefois que les cas traités couvrent l'ensemble des séquences d'événements. Pour vérifier la conformité d'un cas quelconque, il sera alors possible de s'y référer en identifiant la situation et en adaptant l'échelle temporelle.

2.5 Contraintes du projet

La section qui suit traite de la gestion du projet. Il s'agit de préciser la planification du projet avec des divers livrables à fournir à des dates précises. La figure (**??**) ci-dessous est le plan de développement du projet. Les losanges bleus sont les jalons à fournir. La conception de cet IP a débuté le 5 octobre 2022, c'est-à-dire à la 40$^{\text{ème}}$ semaine de l'année. Le rendu de l'IP et du rapport de conception s'effectuera la semaine 49.

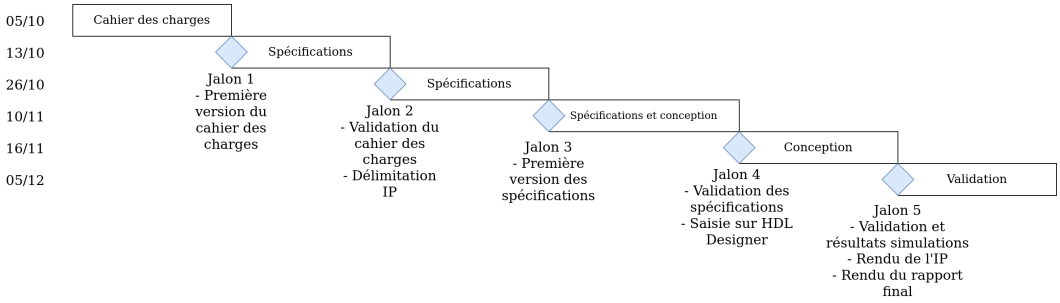

Figure 7: Planification du projet avec ses jalons

Également, le projet est contraint d'un point de vue ressources disponibles. La liste suivante présente les ressources attribuées pour la conception de ce contrôleur d'interruptions.

- 2 concepteurs
- Outils EDA de Mentor Graphics (propriété de Siemens EDA)

- Carte d'évaluation ZYNQ-7 basé sur un FPGA Zynq-7000

3 Spécifications

3.1 Caractérisation de l'environnement

Il s'agit dans cette partie de caractériser l'environnement c'est-à-dire de d'identifier les entités interagissant avec le circuit à concevoir et décrire leur évolution à l'aide d'automates. L'environnement du circuit à concevoir est constitué de trois entités :

- Le processeur
- Le contrôleur mémoire fournissant le signal de sélection nCS_IT.
- Les 15 entités informant au contrôleur d'interruptions la présence d'une interruption. Ce groupe d'entités est appelé "source d'interruptions".

L'ensemble processeur + contrôleur mémoire peut configurer le contrôleur d'interruptions et opérer des écritures et lectures au sein de ses registres. Cet ensemble sera par la suite nommé "système à microprocesseur". L'entité source d'interruptions envoie au circuit à concevoir la présence d'une interruption à traiter.

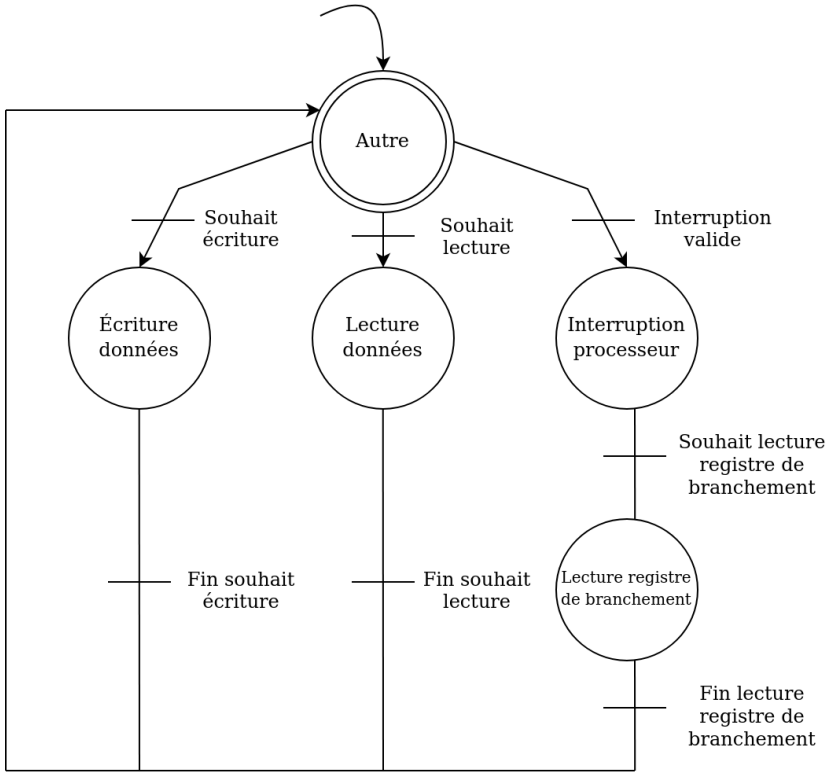

Figure 8: Comportement du système à microprocesseur

La figure ci-dessus représente le comportement de l'entité système à microprocesseur. Celui-ci possède trois cycles de fonctionnement. Le processeur peut opérer des cycles de lecture et d'écriture dans les registres du contrôleur d'interruptions. Le souhait d'écriture ou de lecture s'effectue par la mise à l'état bas du signal nCS_IT. La fin de ces cycles est notifiée par la mise à l'état haut du signal nAS. Le terme "données" peut signifier la valeur des registres de configuration et également l'adresse de branchements.

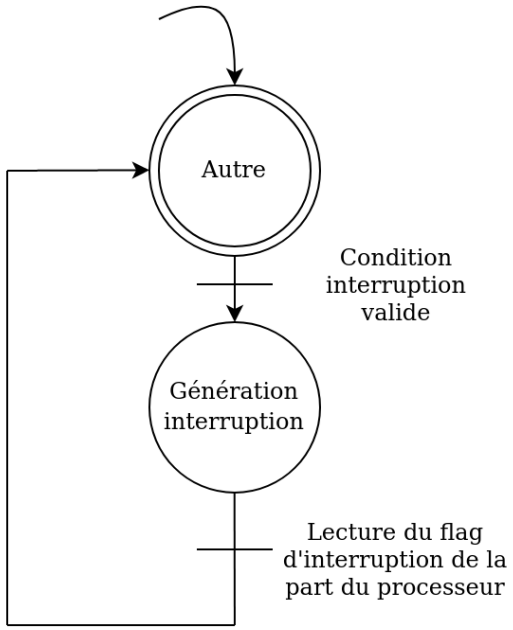

Figure 9: Comportement de l'entité source d'interruptions

La figure ci-dessus est l'automate de l'entité source d'interruptions. Lorsque la condition d'interruption est valide, le périphérique génère un signal en direction du contrôleur d'interruptions. Ce signal reste actif jusqu'à la lecture du flag d'interruption de la part du processeur.

3.2 Entrées et sorties du composant

La caractérisation de l'environnement sous forme d'automates donne les relations d'entrées et sorties du circuit à concevoir avec les diverses entités. Il est alors possible de présenter de manière structurelle le circuit à concevoir et les entités de l'environnement.

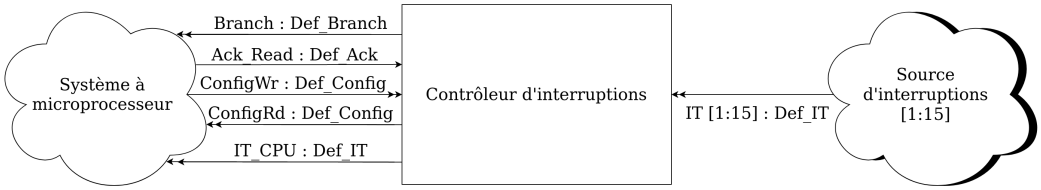

Figure 10: Entrées et sorties du circuit à concevoir

Le tableau ci-dessous récapitule les relations avec leur sens, leur catégorie et leur type.

Entités	Relation	Catégorie	Sens	Type
Système à microprocesseur	Branch	Permanent	Sortie	Def_Branch
	Ack_Read	Évènementiel	Entrée	Def_Ack
	ConfigWr	Permanent	Entrée	Def_ConfigWr
	ConfigRd	Permanent	Sortie	Def_ConfigRd
	IT_CPU	Permanent	Sortie	Def_IT
Source d'interruptions [1:15]	IT [1:15]	Permanent	Entrée	Def_IT

Table 2: Sens et rôle des signaux

La relation Ack_Read est de type Def_Ack, une donnée de type booléen (true : la lecture de l'adresse de branchement est terminée, false sinon). IT[1:15] est un tableau de 15 éléments de type Def_IT, une donnée de type booléen (true : une interruption est reçu de la part de n-ième l'IP, false sinon). IT_CPU est également de type Def_IT (true : une interruption doit être traitée par le CPU, false sinon). Chaque autre relation est une composition de sous-relations comme définie ci-après.

$$\text{Branch} = \text{ID} + \text{@blx}$$

$$\text{ConfigWr} = \text{@blx} + \text{priorité} + \text{masque}$$

$$\text{ConfigRd} = \text{ID} + \text{@blx} + \text{priorité} + \text{masque} + \text{pending} + \text{traitement}$$

ID représente l'identifiant de l'interruption. @blx est la sous-relation qui précise l'adresse de branchement de la sous-routine de l'interruption à traiter. La relation priorité indique la priorité attribuée pour une interruption, masque précise si l'interruption est masquée ou non. En ce qui concerne pending, il s'agit d'une sous-relation qui informe si une interruption est mise en attente pour être traitée ultérieurement. Enfin, traitement prends en compte si une interruption a déjà été traitée ou non avant que celle-ci soit clear par l'IP concernée.

Sous-relation	Type	Donnée
ID	Def_ID	Entier de 0 à 14
@blx	Def_@blx	Entier de 0 à $2^{22} - 1$
priorité	Def_priorité	Entier de 0 à 7 0 : le plus prioritaire 7 : le moins prioritaire
masque	Def_masque	Booléen true : IT masquée (prise en compte) false : non masquée
pending	Def_pending	Booléen true : IT mise en attente false sinon
traitement	Def_traitement	Booléen true : IT à traiter false : IT déjà traitée

Table 3: Sous-relations et type de données

Il est possible d'introduire une relation interne nommée infoIT. Le k-ième élément de infoIT correspond à la k-ième interruption, c'est-à-dire la valeur de ID. Cet indice k varie de 0 à 14 (0-14 représentant les 15 interruptions).

$$\text{infoIT[k]} = \text{ID} + \text{priorité} + \text{masque} + \text{pending} + \text{traitement}$$

3.3 Spécifications fonctionnelles

Les spécifications fonctionnelles comprennent la liste des fonctions du système pour l'application (fonctions externes) et la description du comportement du système et de l'environnement pour ces fonctions [**?**].

La fonction première du contrôleur d'interruption est d'informer le processeur lorsqu'une interruption valide survient. Une interruption valide est définie comme étant non masquée et de niveau suffisant pour être prise en compte. La gestion des niveaux de priorité est la seconde fonction à considérer. Elle est directement induite par la première. Une interruption en cours ne peut être interrompue que par une interruption de niveau strictement supérieur. Lorsque le processeur acquitte la fin du traitement, les interruptions de niveaux inférieurs ou égaux peuvent être à nouveau considérées comme valides. Le contrôleur fournir un niveau de priorité par défaut pour chaque source. La troisième fonction concerne le masquage des sources d'interruption individuellement. Une source masquée ne génèrera pas de signal au processeur. Un événement déjà en cours de traitement par le contrôleur ne pourra être masqué. Si un signal d'interruption est généré alors qu'il est masqué, il est ignoré, mais s'il est démasqué alors qu'il est toujours actif alors il sera pris en compte. (A confirmer) La quatrième fonction principale spécifie le vecteur d'exception. Le contrôleur lorsque qu'il informe le processeur doit immédiatement lui indiquer vers quelle adresse il doit brancher. Il est possible de configurer une adresse par source. Sa configuration doit se faire lorsque le contrôleur est désactivé. Si aucune adresse n'est présente pour une source donnée alors une par défaut sera attribuée.

nom	Description
fn1	Informer le processeur
fn2	Gérer les niveaux de priorité
fn3	Masquer individuellement les sources
fn4	Configurer le vecteur d'exception

Table 4: Synthèse des spécifications fonctionnelles

Le tableau **??** synthétise les spécifications fonctionnelles. Il permet de contrôler que chaque point est bien traité tout au long de la conception. Il n'est cependant pas précis et devra être traité avec la description détaillée ci-dessus.

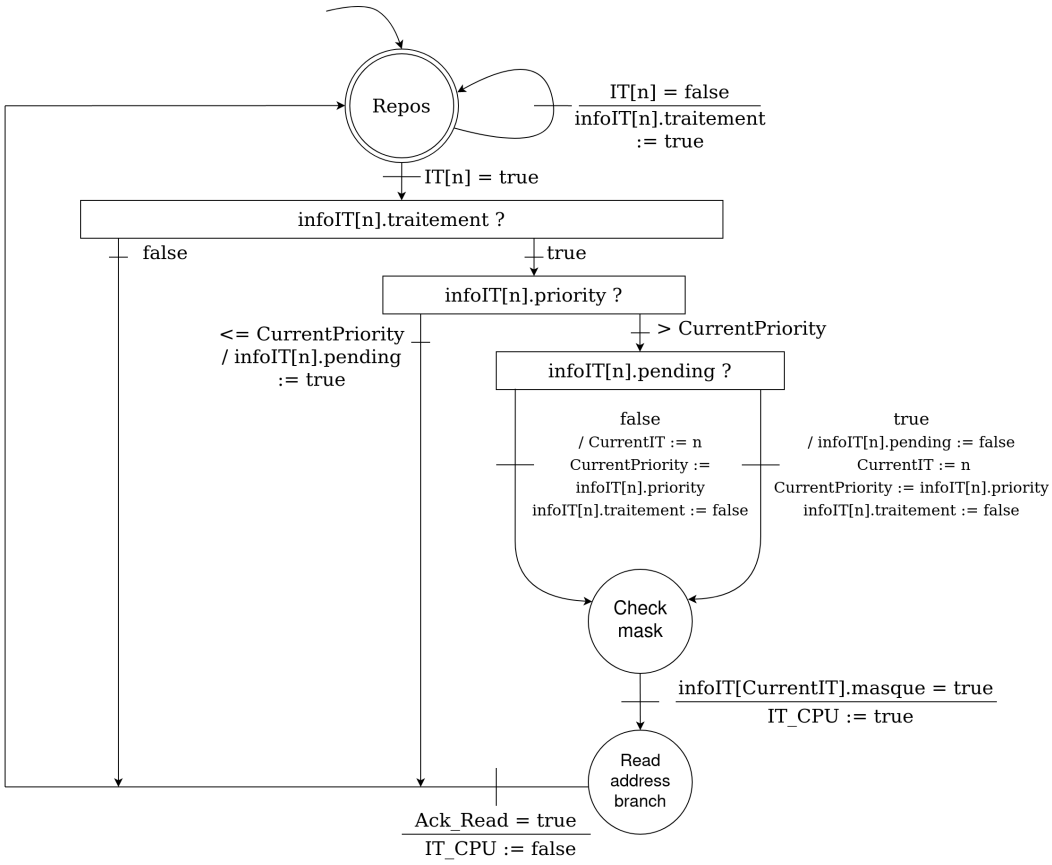

Figure 11: Automate du contrôleur d'interruptions

ID	Interruptions correspondantes	Adresses de branchement par défaut
0	nIT_ext0	
1	nIT_ext1	
2	nIT_ext2	
3	nIT_ext3	
4	nIT_RTC	
5	nIT_TC_PWM	
6	nIT_pos_vit	
7	nIT_FFTA	
8	nIT_NNA	
9	nIT_SPI	
10	nIT_PCI	
11	nIT_UART	
12	nIT_I2C	
13	nIT_CAN	
14	nIT_DMA	

Table 5: Vecteurs d'interruptions

3.4 Spécification des registres

Adresse	Nom	Type	Reset	Description
0x00000000	CTRL_IT_EN	RW	0x0000	Registre activation contrôlleur d'IT
0x00000002	CTRL_IT_MSQ	RW	0x0000	Registre de masquage
0x00000004	CTRL_IT_PEND	R	0x0000	Registre IT suspendues (pending)
0x00000006-0x00000008	CTRL_IT_BRA	R	0x0000	Registre de branchement
0x0000000A-0x0000000C	CTRL_IT_ADDR_0	RW	0x0000	Adresses de branchement ch 0
0x0000000E-0x00000010	CTRL_IT_ADDR_1	RW	0x0000	Adresses de branchement ch 1
0x00000012-0x000000014	CTRL_IT_ADDR_2	RW	0x0000	Adresses de branchement ch 2
0x00000016-0x000000018	CTRL_IT_ADDR_3	RW	0x0000	Adresses de branchement ch 3
0x0000001A-0x0000001C	CTRL_IT_ADDR_4	RW	0x0000	Adresses de branchement ch 4
0x0000001E-0x00000020	CTRL_IT_ADDR_5	RW	0x0000	Adresses de branchement ch 5
0x00000022-0x000000024	CTRL_IT_ADDR_6	RW	0x0000	Adresses de branchement ch 6
0x00000026-0x00000028	CTRL_IT_ADDR_7	RW	0x0000	Adresses de branchement ch 7
0x0000002A-0x0000002C	CTRL_IT_ADDR_8	RW	0x0000	Adresses de branchement ch 8
0x0000002E-0x00000030	CTRL_IT_ADDR_9	RW	0x0000	Adresses de branchement ch 9
0x00000032-	CTRL_IT_ADDR_10	RW	0x0000	Adresses de branchement ch 10
0x00000034-0x00000036	CTRL_IT_ADDR_11	RW	0x0000	Adresses de branchement ch 11
0x00000038-0x0000003A	CTRL_IT_ADDR_12	RW	0x0000	Adresses de branchement ch 12
0x0000003C-0x0000003E	CTRL_IT_ADDR_13	RW	0x0000	Adresses de branchement ch 13
0x00000040-0x00000042	CTRL_IT_ADDR_14	RW	0x0000	Adresses de branchement ch 14
0x00000044	CTRL_IT_PRIO_0_1	RW	0x0000	Registre des priorités
0x00000046	CTRL_IT_PRIO_2_3	RW	0x0000	Registre des priorités
0x00000048	CTRL_IT_PRIO_4_5	RW	0x0000	Registre des priorités
0x0000004A	CTRL_IT_PRIO_6_7	RW	0x0000	Registre des priorités
0x0000004C	CTRL_IT_PRIO_8_9	RW	0x0000	Registre des priorités
0x0000004E	CTRL_IT_PRIO_10_11	RW	0x0000	Registre des priorités
0x00000050	CTRL_IT_PRIO_12_13	RW	0x0000	Registre des priorités
0x00000052	CTRL_IT_PRIO_14	RW	0x0000	Registre des priorités

Table 6: Organisation des registres du contrôleur d'interruption (vue globale)

Bit	Nom	Fonction		
[0]	Enable	Active ou lit l'état d'activation du périphérique		
		Lecture	0	Le périphérique est désactivé
			1	Le périphérique est activé
		Écriture	0	Désactivation du périphérique
			1	Activation du périphérique
[15 : 1]	-	Réservés		

Table 7: Détails du registre CTRL_IT_EN

Bit	Nom	Fonction		
[14 : 0]	Pending	Lit l'état de suspension (pending) de la source		
		Lecture	0	La source est désactivé
			1	La source est activé
		Écriture	0	No effect
			1	No effect
[15]	-	Réservé		

Table 8: Détails du registre CTRL_IT_PEND

Bit	Nom	Fonction
[15 : 0]	BRA_LSB	Lit 16 LSB adresse de branchement
[7 : 0]	BRA_MSB	Lit 8 MSB adresse de branchement
[15 : 8]	-	Réservés

Table 9: Détails du registre CTRL_IT_BRA

Bit	Nom	Fonction
[15 : 0]	ADDR_LSB	Lit et écrit 16 LSB adresse de branchement
[7 : 0]	ADDR_MSB	Lit et écrit 8 MSB adresse de branchement
[15 : 8]	-	Réservés

Table 10: Détails du registre CTRL_IT_ADDR_N

Bit	Nom	Fonction
[2 : 0]	PRIO_N	Lit et écrit le niveau de priorité de N
[7 : 3]	-	Réservés
[10 : 8]	PRIO_N+1	Lit et écrit le niveau de priorité de N+1
[15 : 11]	-	Réservés

Table 11: Détails du registre CTRL_IT_PRIO_N_N+1

Il est choisi de placer les priorités sur les 3 premiers bits de chaque octet. Cela permet une manipulation aisée en C et optimal pour un processeur possédant des instructions de

manipulation d'octets. Elle n'est cependant pas optimale pour l'occupation mémoire qui passe au second plan.

3.5 Écriture des procédures de base pour l'emploi du circuit

```
1  #include <stdint.h>
2  #define CTRL_IT_BASE_ADDR 0xF0F0F0
3
4  typedef enum {
5      nIT_ext0 = 0, nIT_ext1, nIT_ext2, nIT_ext3, nIT_RTC, nIT_TC_PWM,
6      nIT_pos_vit, nIT_FFTA, nIT_NNA, nIT_SPI, nIT_PCI, nIT_UART,
7      nIT_I2C, nIT_CAN, nIT_DMA, NB_IT
8  }ID_IT_t;
9
10 #pragma pack(push, 4)
11 typedef struct {
12     uint8_t ch : 3;
13     uint8_t RESERVED : 5;
14 }CTRL_IT_PRIO_t;
15 #pragma pack(pop)
16
17 #pragma pack(push, 53)
18 typedef struct {
19     uint16_t CTRL_IT_EN;    /* size : 2 */
20     uint16_t CTRL_IT_MSK;   /* size : 2 */
21     uint16_t CTRL_IT_PEND;  /* size : 2 */
22     void (*handler)(void);  /* size : 2 */
23     /* array of function pointer */
24     void (*CTRL_IT_ADDR[NB_IT])(void); /* size : 2x15=30 */
25     CTRL_IT_PRIO_t prio[NB_IT]; /* size : 1x15 */
26 }CTRL_IT_t;
27 #pragma pack(pop)
28
29 CTRL_IT_t * CTRL_IT = (CTRL_IT_t * ) CTRL_IT_BASE_ADDR;
```

Le code ci-dessus défini la structure du contrôleur IT en langage C. Le résultat est le type `CTRL_IT_t`. Une instance globale est créée et initialisée avec l'adresse de base du périphérique.

```
1  void UART_Handler(void)
2  {
3      /* Do somethink */
4      /* clear UART IT flag */
5  }
6
7  void CTRL_IT_init_UART(void)
8  {
9      /* Disable CTRL IT */
10     CTRL_IT->CTRL_IT_EN = 0x0000;
11     /* Mask the IC channel */
12     CTRL_IT->CTRL_IT_MSK = 1UL << nIT_UART;
13     /* Set handler addr */
14     CTRL_IT->CTRL_IT_ADDR[nIT_UART] = &UART_Handler;
15     /* Enable CTRL IT */
16     CTRL_IT->CTRL_IT_EN = 0x0001;
17 }
```

Les deux procédures ci-dessus présentent un exemple d'initialisation du périphérique pour le canal UART.

3.6 Spécifications opératoires

3.7 Spécifications technologiques

4 Conclusion

Pour conclure, la méthode MCSE apporte un outil complet offrant de nombreux avantages. Le principale étant d'effectuer une bonne structuration de sa conception, mais également de raisonner en faisant abstraction de la technologie utilisée. Cela permet de ne pas se confiner dans une solution technique. Également, la partie spécification est très utile pour dégrossir le cahier des charges ou pour réduire la complexité de conceptions lourdes, d'éviter les erreurs. Cependant la méthode MCSE n'est pas applicable à des conceptions très basiques et simples. C'est en effet plus efficace de s'abstenir de cet outil car celui-ci est chronophage, et il est parfois nécessaire de réitérer l'ensemble du processus lorsqu'un problème de conception est identifié. Entre autres, l'outil qu'est la méthode MCSE est puissant mais difficile à maîtriser car celui-ci demande une grande expérience.

CPU
IP
TTL

Acronyms

CPU Central Processing Unit 6, 9, 22

IP Intelectual Property 6, 22

TTL Transistor Transistor Logic 9, 22

[**?**]

References

[1] J.P. Calvez. *Spécification et conception des systèmes - une méthodologie*. 1991.

[2] J.P. Calvez. *Spécification et conception des systèmes - étude de cas*. 1991.

[3] STMicroelectronics. Reference manual rm0376. Technical report, STMicroelectronics, February 2022.

5 Appendix