PROJET DE SYNTHESE

Intégration d’un micro-contrôleur dans un ASIC

Sylvain GARNIER (MICROCHIP Nantes)

Ingénieur 3ème année (EOC) **2024-2025**

10 Bd Jeanneteau - CS 90717 - 49107 ANGERS Cedex 2 - Tél : 02 41 86 67 67 - Fax : 02 41 87 99 27

Table des matières

1 Introduction

1.1 Objectif du projet

Amener les étudiants à mettre en œuvre les enseignements de micro-électronique et d'informatique embarquée dispensés en I1, I2 et I3 dans le cadre de la réalisation d'un projet industriel (ASIC synthétiseur de fréquence).

1.2 Micro-contrôleur versus logique câblée

Les moyens de réaliser une fonction logique sont :

- La logique câblée, qui permet de réaliser la fonction précise avec un grand degré de liberté quant à son optimisation surface ou temps d'exécution. Toute modification de la fonction entraîne le renouvellement du design.
- La programmation est une autre approche qui permet d'implémenter une fonctionnalité aisément modifiable et réutilisable, et ce très rapidement, à un niveau d'abstraction élevé.

1.3 Objectif de réalisation

Il s'agit de réaliser un circuit capable de contrôler un convertisseur numérique-analogique (CAN) externe. L'application choisie est un générateur de sinusoïde de fréquence F=1Khz. Les moyens à votre disposition sont :

- les bases d'un micro-contrôleur dont il va falloir développer des instructions
- un compilateur ANSI C+assembleur 51 (SDCC)
- simulateur RTL
- synthétiseur logique (RTL Compiler (RC) de CADENCE)
- Simulateur analogique ELDO couplé au simulateur RTL (simulation mixte avec la suite Advance MS)

1.4 Planning

Séance 1	**AM**	Présentation du projet Éléments de cours Présentation de l'architecture du 8051 Définition du besoin applicatif Modélisation haut niveau de l'appli
	PM	Définition de l'architecture du circuit Prototypage de l'application sur cible Analyse des ressources à implémenter Implémentation du micro-contrôleur Co-simulation RTL+application
Séance 2	**AM**	Spécification et conception d'un périphérique TIMER Intégration du périphérique Vérification
	PM	Ré-écriture de l'application (scrutation, interruption, IDLE)
RTL		

Séance 3	**AM**	Éléments de cours (Synthèse logique) Synthèse logique du TIMER Analyse des résultats de synthèse (surface, timings) Analyse statique de timings Synthèse logique du circuit
	PM	Simulations NTL 0-delay Simulations NTL rétro-annotées
Séance 4	**AM**	Analyse statique de timing (setup) Optimisation du circuit en fréquence
	PM	Optimisation du circuit en consommation (synthèse low-power)
NTL		

Séance 5	**AM**	Introduction à la simulation mixte Mise en place de l'environnement de simulation mixte (ADMS) Écriture d'un modèle VHDL-AMS Simulation mixte VDHL+VHDLAMS
	PM	Simulation mixte avec netlist spice Modélisation d'un R2R DAC et simulation
MIXED-SIM		

1.5 Définitions

ASIC (Application-specific integrated circuit)
Circuit intégré conçu pour répondre à un besoin particulier à opposer aux circuits généralistes (Standard products).

ASSP (Application specific standard product) :
Un ASSP est un circuit intégré comportant des fonctions spécifiques qui adressent un large marché contrairement à un ASIC qui ne répond qu'à une application et souvent à un seul client. Un ASSP est souvent conçu autour d'une architecture standard, ainsi, si l'on n'utilise pas les fonctions spécifiques on peut aussi l'utiliser comme un produit standard.

Micro-processeur:
Un microprocesseur est un processeur dont les composants ont été suffisamment miniaturisés pour que l'ensemble du processeur puisse tenir sur un seul circuit intégré. Fonctionnellement, le processeur est la partie d'un ordinateur qui exécute les opérations arithmétiques et logiques contenues dans les programmes qui composent la partie logicielle de cet ordinateur.

Micro-contrôleur:
Selon un arrêté français du 14 septembre 1990 relatif à la terminologie des composants électroniques : « Circuit intégré comprenant essentiellement un microprocesseur, ses mémoires, et des éléments personnalisés selon l'application ».

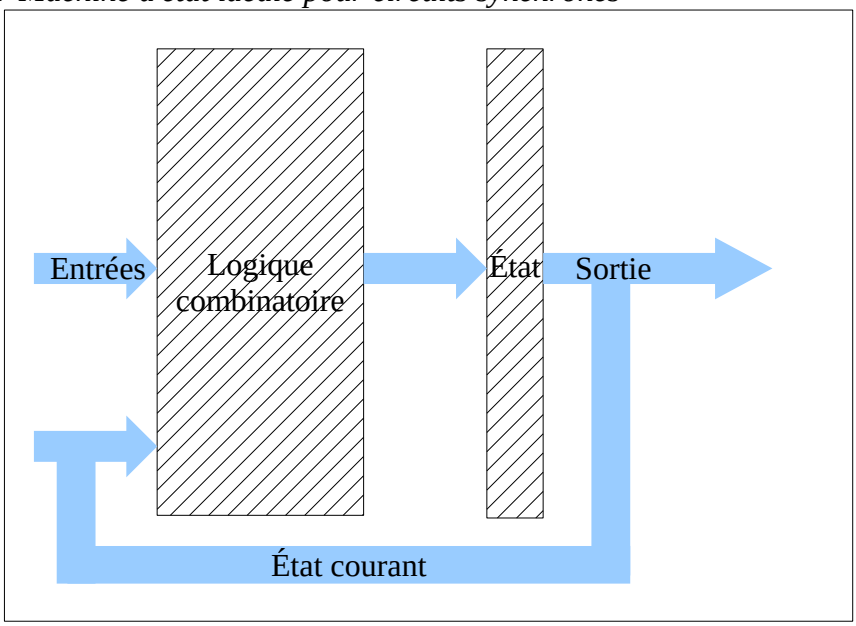
Un micro-contrôleur peut comprendre :
- de la mémoire morte dite ROM (principalement pour stocker le programme) ;
- de la mémoire vive dite RAM (principalement pour stocker les variables) ;
- des périphériques (principalement pour interagir avec le monde extérieur), ex:
    - les convertisseurs Analogique/Numérique
    - les convertisseurs Numérique/Analogique
    - les générateurs de signaux à modulation de largeur d'impulsion (PWM) ;
    - les timers (compteurs de temps ou d'événements) ;
    - les comparateurs (comparent deux tensions électriques) ;
    - les contrôleurs de bus (UART, IIC) ;
- une horloge pour le cadencer

1.6 Éléments de cours (rappels de cours)

1.6.1 Codage des machines d'états

Les machines d'état les plus connues sont celles de Moore et de Mealy. Elles présentent toutes deux des avantages et des inconvénients notamment de par le fait que leurs sorties sont combinatoires. Ce qui est recommandé est représenté sur la figure suivante (toutes les sorties sont issues de bascules) :

1 Machine d'état idéale pour circuits synchrones

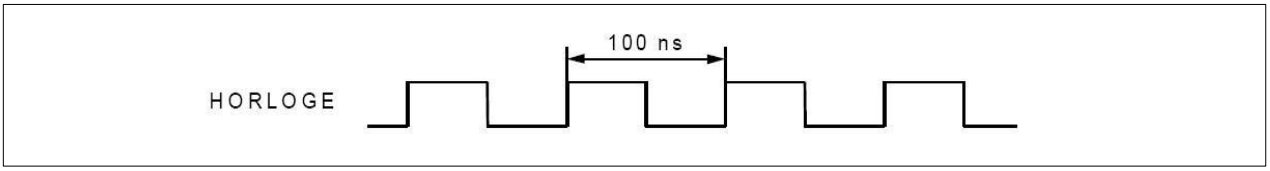


1.6.2 Synthétisable/Comportemental

Description comportementale:
Voici l'exemple de la description en VHDL d'une horloge à 10Mhz :

```
Horloge <= not Horloge after 50 ns ;
```

2 Chronogramme d'une horloge, f=10Mhz, rapport cyclique=50%



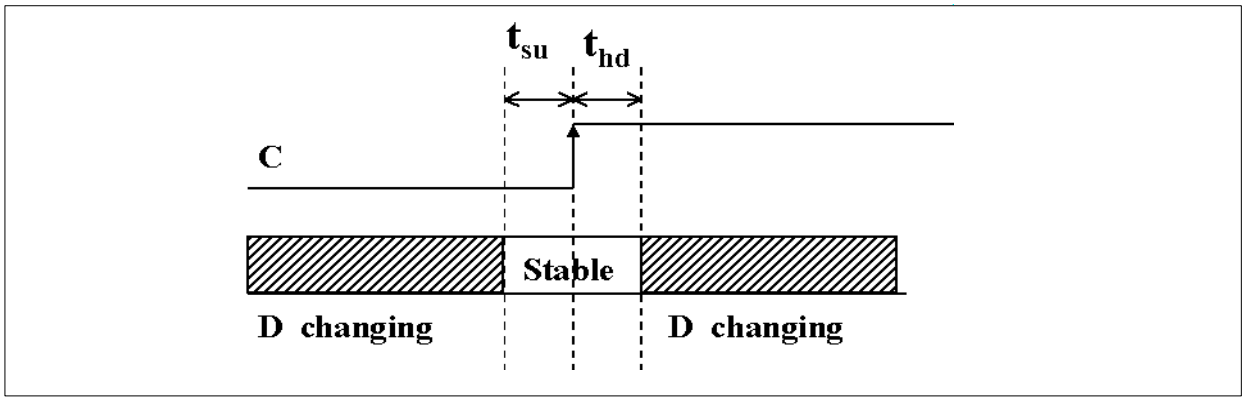
Cette description est correcte en VHDL, mais n'est pas synthétisable

Description synthétisable:
logique séquentielle
logique combinatoire

1.6.3 Contraintes liées aux temps de propagation

- Temps de setup/hold d'une DFF

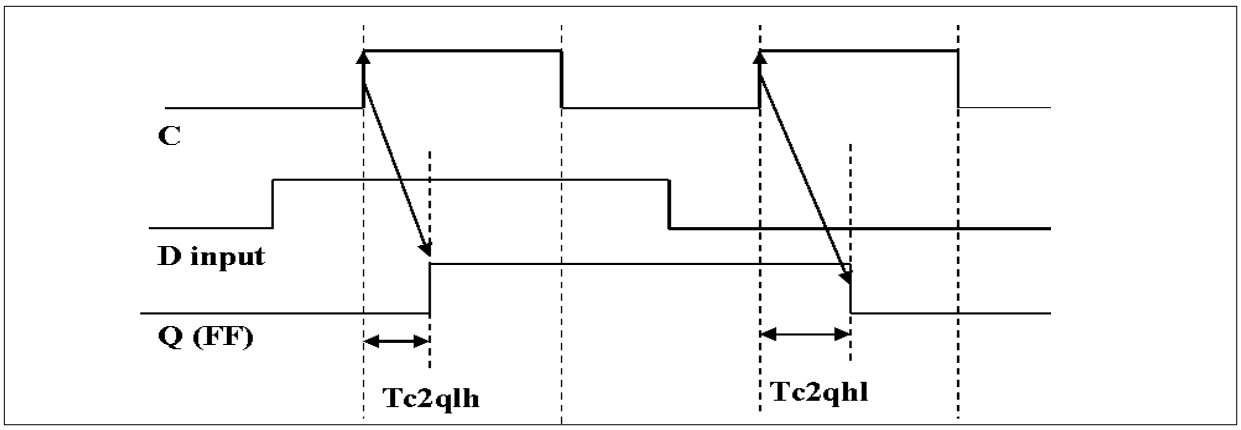
3 Figure: Setup/hold d'une DFF



L'entrée D de la DFF doit être stable t_{su} unités de temps avant le front montant et t_{hd} après. Le comportement de la flip-flop n'est pas garanti sinon.

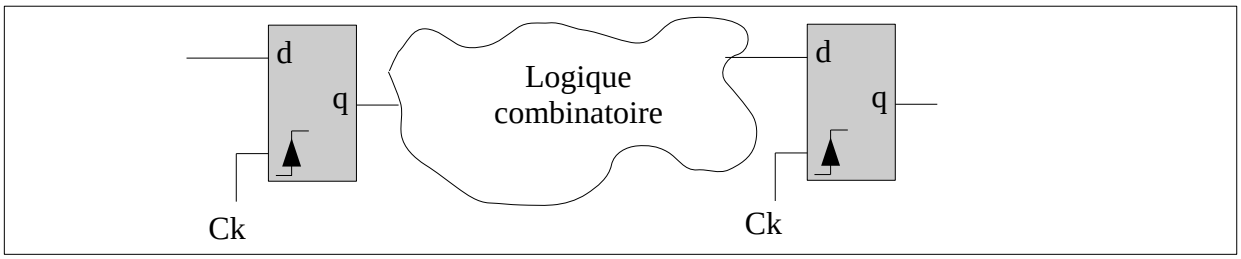
- Temps intrinsèque de propagation d'une DFF

4 Figure: Temps de propagation intrinsèque d'une DFF



- Fréquence max d'un circuit

5 Figure: Fréquence max d'un circuit



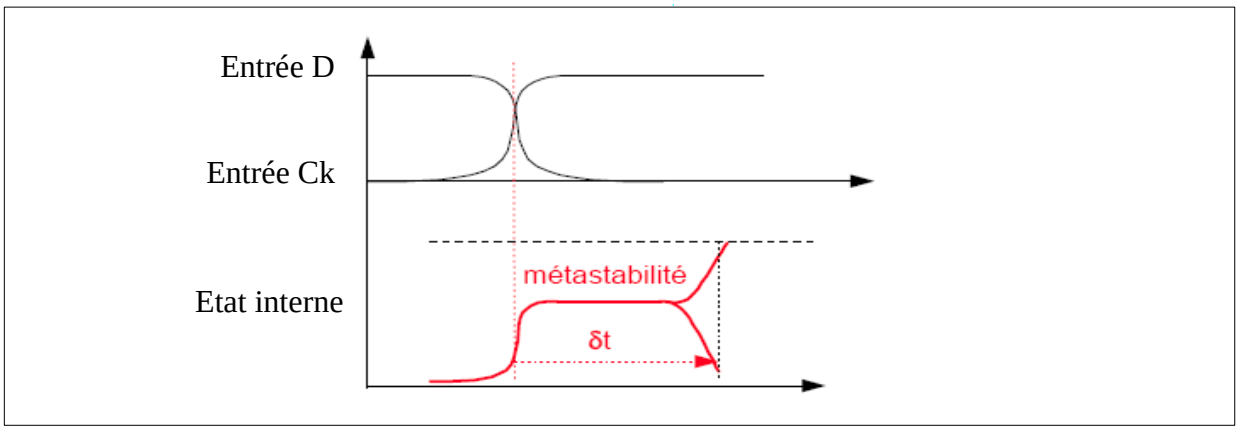
Soit E l'ensemble des chemins entre les sorties Q et entrées D des bascules appartenant à un même domaine d'horloge dans un circuit donné. La fréquence max du système dans un corner donné est donnée par le temps de propagation du chemin le plus long de E.

$$fmax=1/(T(max(E))+tsu+max(Tc2qlh, Tc2qhl))$$

1.6.4 Méta-stabilité

Le fonctionnement "normal" d'une bascule est obtenu lorsque ses entrées sont bien établies au moment de la transition du signal d'horloge. Il peut toutefois arriver que dans certaines applications une entrée de la bascule varie au moment précis de la transition du signal d'horloge. Si la relation temporelle entre ces deux transitions est suffisamment précise, la bascule peut basculer incomplètement car elle ne dispose pas de suffisamment d'énergie pour basculer complètement. Elle n'atteint alors qu'une position intermédiaire que nous appellerons ½. Ceci ne serait pas très gênant si le retour à une position logique (1 ou 0) ne générait pas une transition asynchrone de sa sortie qui risque de mettre aussi en état métastable la bascule suivante.

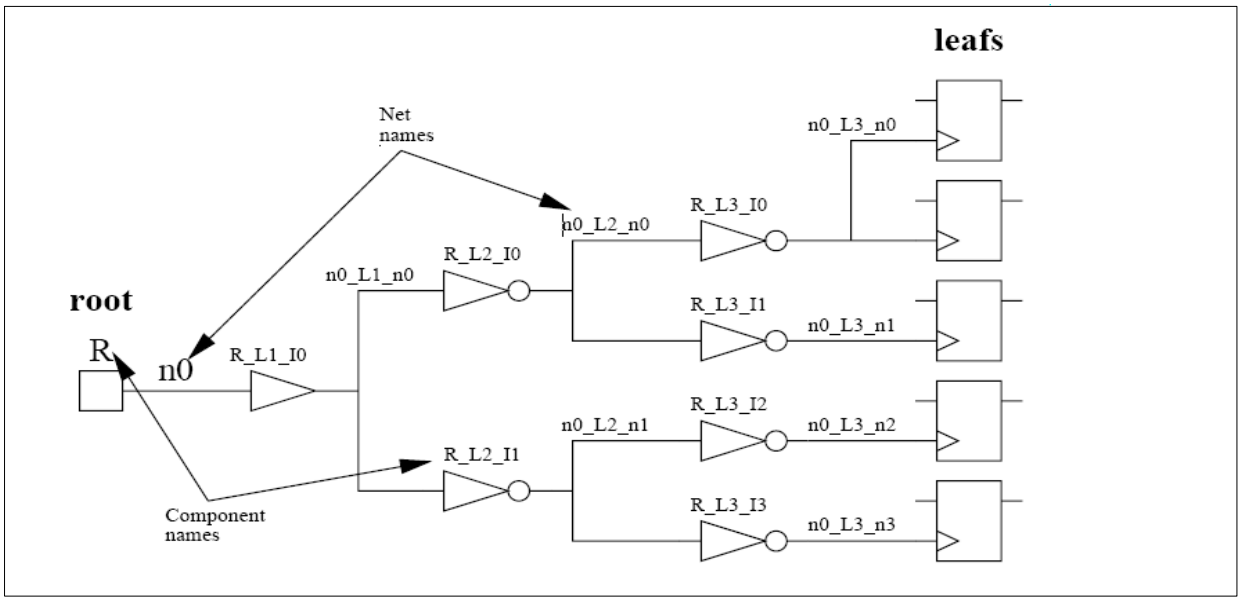
6 Métastabilité



La largeur de la relation temporelle qui peut mettre une bascule en état métastable est très faible(de l'ordre d'une fraction de pS). La probabilité d'occurrence de la métastabilité pour une bascule, qui reçoit des signaux complètement désynchronisés, est donc très faible (10^{-12} à 10^{-14}). Les fréquences de fonctionnement étant très élevées, la probabilité est non négligeable.

1.6.5 Arbres d'horloge (consommation, bruit numérique)

7 Exemple d'arbre d'horloge à 3 niveaux



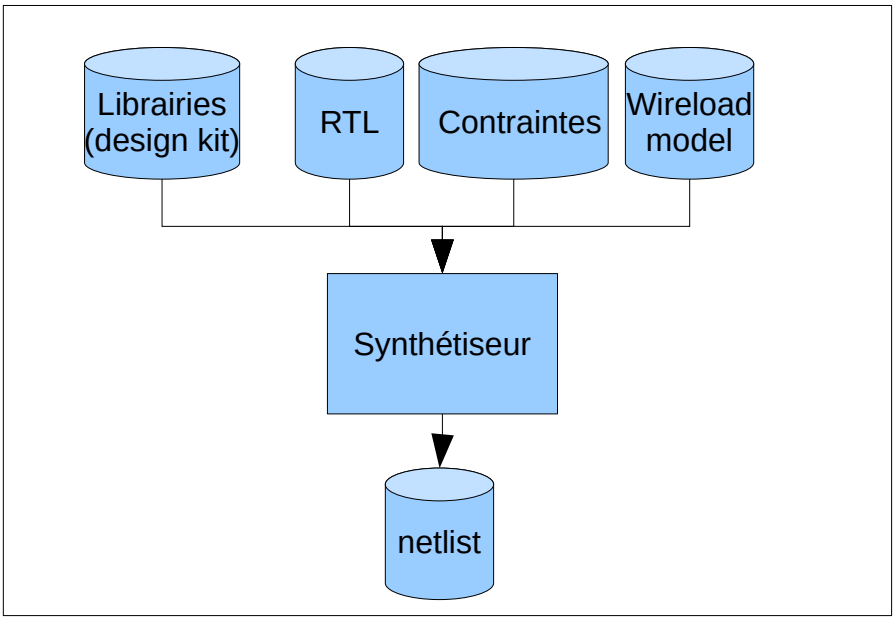
1.6.6 Synthèse logique

1.6.6.1 Général

Les résultats d'une synthèse se mesurent selon 3 critères :

- Surface du design
- Vitesse
- Consommation

8 Diagramme de flux basique d'un synthétiseur logique



1.6.6.2 Le synthétiseur

L'outil de synthèse d'un circuit synchrone procède généralement en plusieurs étapes :

- il va d'abord transformer la description RTL en éléments logiques combinatoires et séquentiels génériques (indépendamment de la bibliothèque cible) suivant des algorithmes mathématiques.
- il va ensuite remplacer les éléments logiques génériques par ceux issus de la bibliothèque cible. Pour cela, il choisit les éléments logiques respectant les contraintes de temps et d'espace donnés par l'utilisateur. Des calculs d'analyse de délais sont alors réalisés sur tous les chemins logiques du circuit afin de s'assurer qu'ils respectent les contraintes de temps (fréquence de fonctionnement du circuit). Si les résultats ne sont pas concluants, l'outil essaye d'utiliser d'autres portes disponibles dans la bibliothèque pour arriver au résultat souhaité. Il est ainsi courant dans une bibliothèque d'avoir de nombreuses portes réalisant la même fonction logique mais avec des tailles et des « drive » différentes.
- Lorsque les contraintes de temps sont remplies, l'outil de synthèse dispose de certaines marges de temps sur certains chemins. Il peut alors remplacer certaines portes par d'autres moins gourmandes en consommation et en taille silicium tout en continuant à respecter les contraintes de temps.

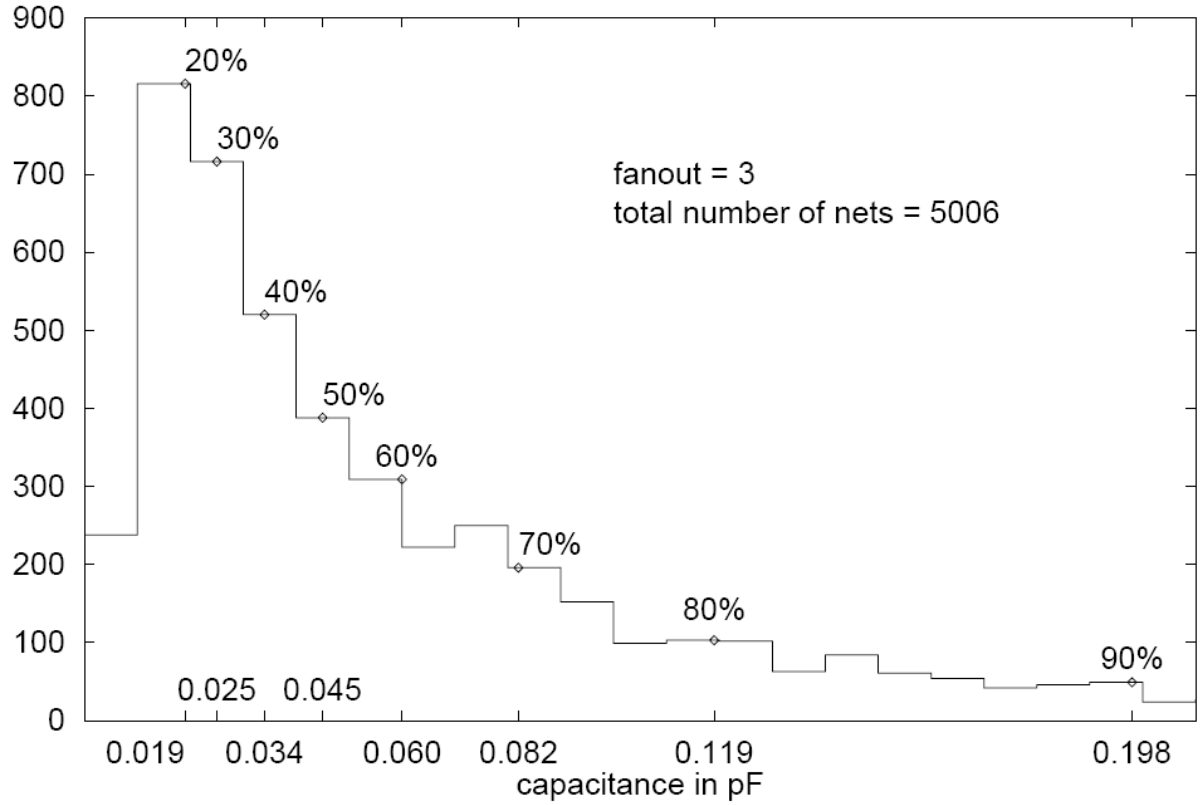
1.6.6.3 Les entrées de la synthèse logique

La qualité du résultat de la synthèse dépend de :

- Librairies bien caractérisées
- RTL bien conçu
    - partage des ressources (ex: ALU)
    - choix de l'encodage des machines d'état (ex: binary, gray, one-hot)
    - Complexité de la logique combinatoire réaliste par rapport aux objectifs de vitesse
- Contraintes pertinentes
    - fréquence d'horloge demandée réaliste
    - Timings Exceptions, ex :
        - « false-paths »
        - « multi-cycle paths »
- Données physiques réalistes
    - Synthèse « topographique »avec un floorplan réaliste
    - ou choix du « wireload model » réaliste

  Le « wireload model » est très important en synthèse et souvent sous estimé quant à son impact sur le résultat. Le « wireload » est utilisé par le synthétiseur pour déterminer les caractéristiques des interconnections en l'absence de données physiques de placement. Pour un net avec un « fanout » donné, le « wireload model » donne une correspondance en termes de (R, C) et de surface du net. Un « wireloal model » est issu d'une analyse statistique réalisée sur un ensemble de circuits existants. Pour un « fanout » donné, on fait correspondre une unique capacitance.

9 Exemple de distribution en nombre de nets fonction de la capacité:

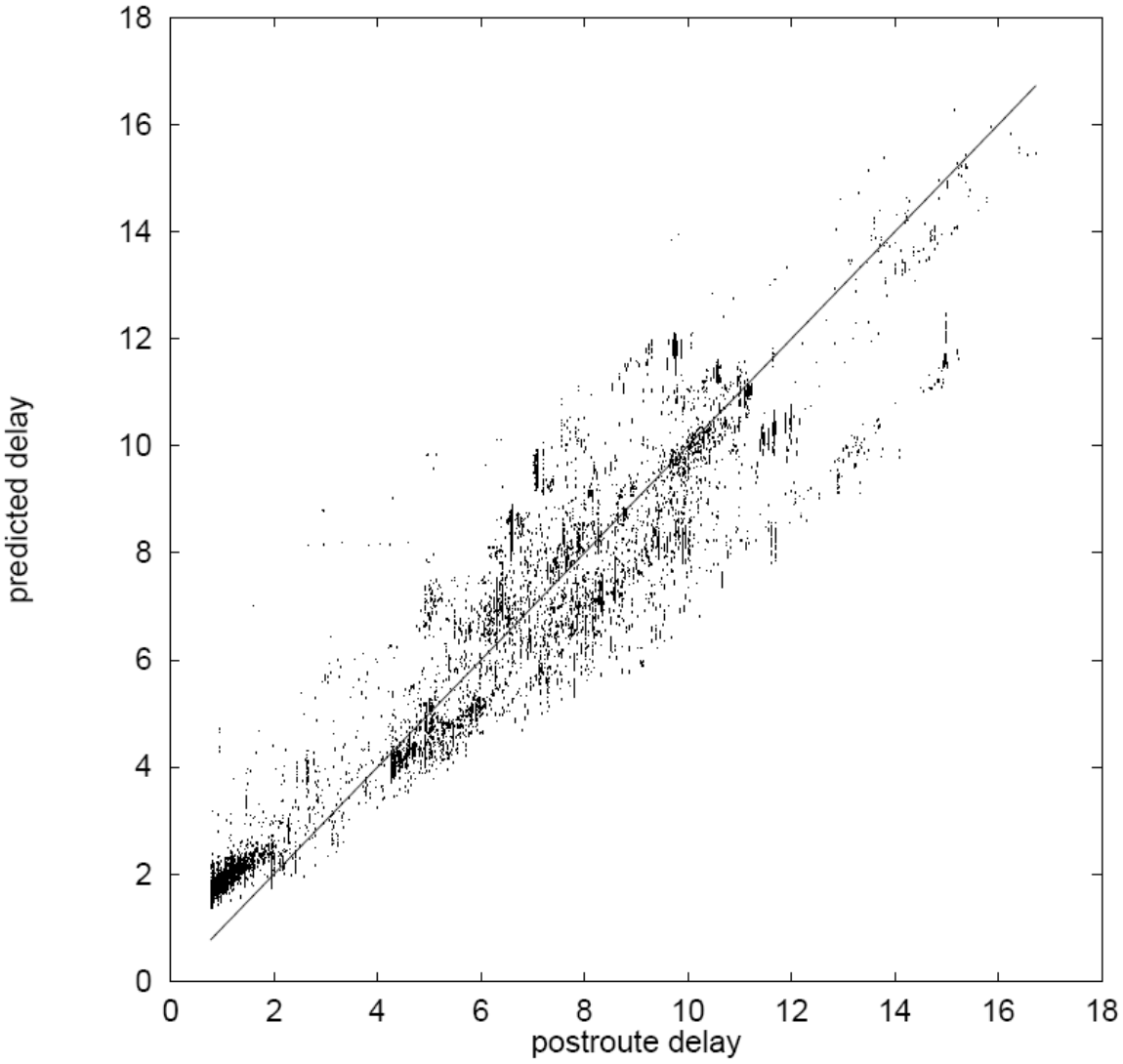


Si l'on veut un « wireload model » conservatif, on prendra le décile 90%, c'est à dire que 90% des nets parmi les échantillons analysés ont une capacité inférieure à la valeur correspondante.

Définition : le fanout (ou capacité de branchement) désigne le nombre maximal de charges ou d'entrées logiques que la sortie d'une porte logique ou d'un circuit peut piloter tout en respectant les contraintes de performance, telles que le temps de propagation, la consommation d'énergie, et la fiabilité des signaux.

Si une porte NAND peut piloter jusqu'à 4 entrées logiques d'autres portes tout en respectant les contraintes temporelles, on dira que cette porte a un fanout de 4.

10 Prédiction lors de la synthèse versus résultat post layout (cf. Toshiba, "Toshiba/Synopsys Links to Layout Methodology," in SNUG 1996 Proceedings.)



Un point représente un « iopath », la diagonale représente une prédiction exacte par rapport au délai constaté post-layout.

La qualité de la netlist dépend également de l'outil de synthèse:

- Capacité à inférer des opérateurs arithmétiques et à choisir et changer leur architecture en fonction des contraintes de timing pendant le processus de synthèse
- Capacité à optimiser les fonctions logique, à factoriser les ressources
- Capacité à dupliquer la logique en fonction des charges

1.6.7 POR (power-on reset generator)

Un générateur de POR est un bloc analogique embarqué sur des micro-contrôleurs pour générer un reset à la mise sous tension du circuit. Cela permet de faire démarrer le circuit dans un état connu. Un POR simple est composé d'un RC qui se charge en même temps que l'alimentation s'établit. Un mécanisme à base de « trigger de Schmitt » est utilisé pour générer une impulsion.

1.6.8 BOD (brown-out detector) ou superviseur de tension

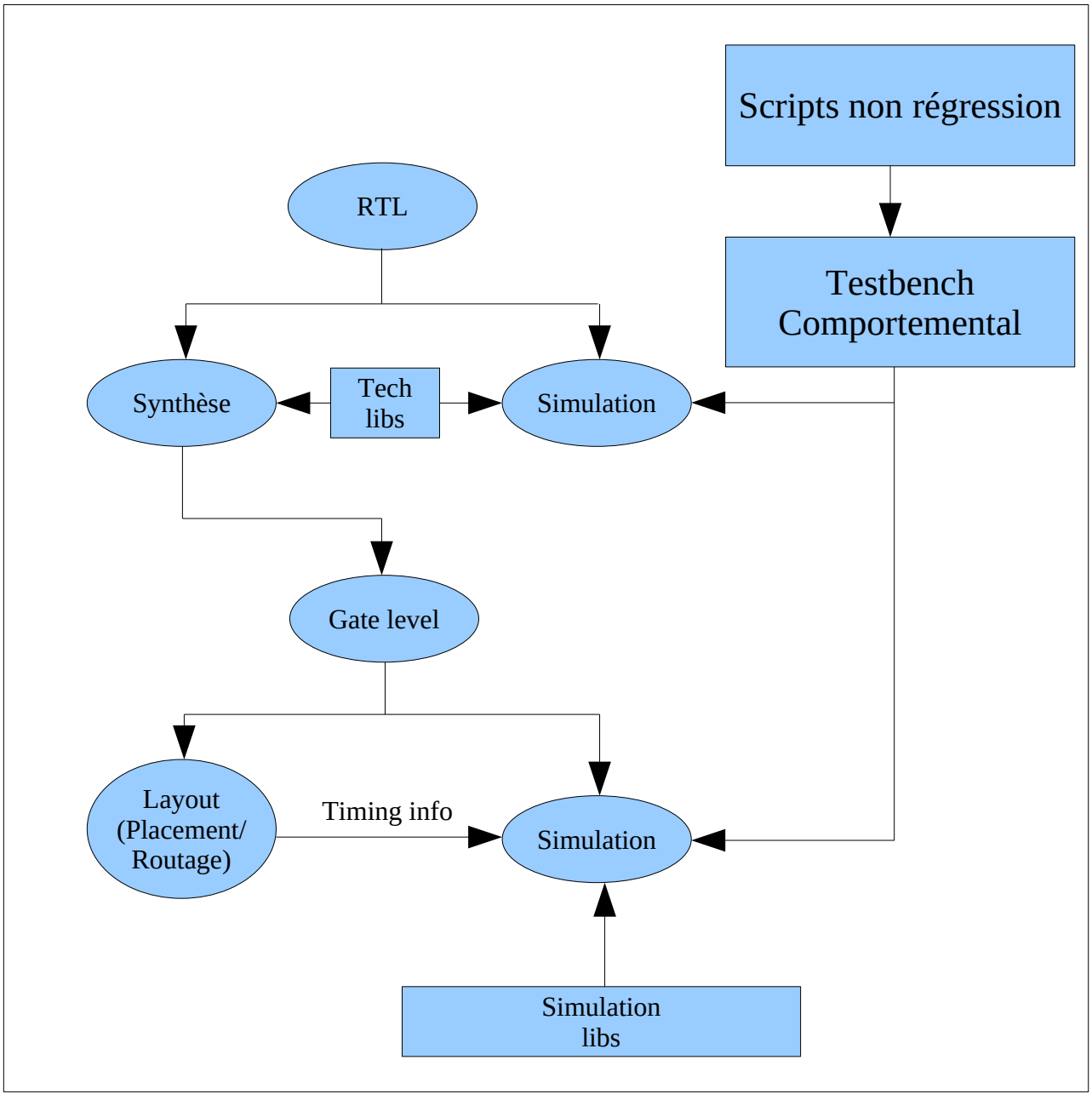
Un BOD ou superviseur de tension permet de surveiller l'alimentation en générant une impulsion lorsque la tension descend en dessous d'un seuil critique. Cela garantit l'intégrité des périphériques de stockage mémoire tels que les flashs qui pourraient être corrompus par l'exécution d'un enchaînement d'instructions non souhaité provoqué par une corruption de l'état interne du système. Dans le jargon applicatif on dit que « le micro jardine ».

2 Aspects Hardware

2.1 Flow de simulation

Le flow est présenté sur le diagramme suivant. Les simulations rétro-annotées pourront se faire avec le fichier de timing issu du placement routage si celui-ci est mis en œuvre. Dans le cas contraire on extraira les timings à l'aide du synthétiseur en lui injectant un « wireload model ».

11 Diagramme de flow



2.2 Vue utilisateur du 8051

2.2.1 Terminologie

SFR (Special function register):

Accessibles dans le plan mémoire direct, ils sont vus comme de la mémoire mais ce sont des registres mappés. Ces registres sont des registres des interfaces des périphériques.

A, ACC: Registre système « Accumulateur »

2.2.2 Jeu d'instruction

Voir section 11.2 « Jeu d'instruction complet»page 61.

Le jeu d'instruction du 8051 est non orthogonal. On dit qu'un jeu d'instruction est orthogonal lorsque toutes les instructions peuvent s'appliquer à tous les types de données. Cela simplifie l'étape de compilation car il n'y a pas de cas particuliers à traiter.

2.2.3 Registres systèmes

- PSW: program status word. bit addressable. (8 bits)

CY	AC	F0	RS1	RS0	OV	F1	P

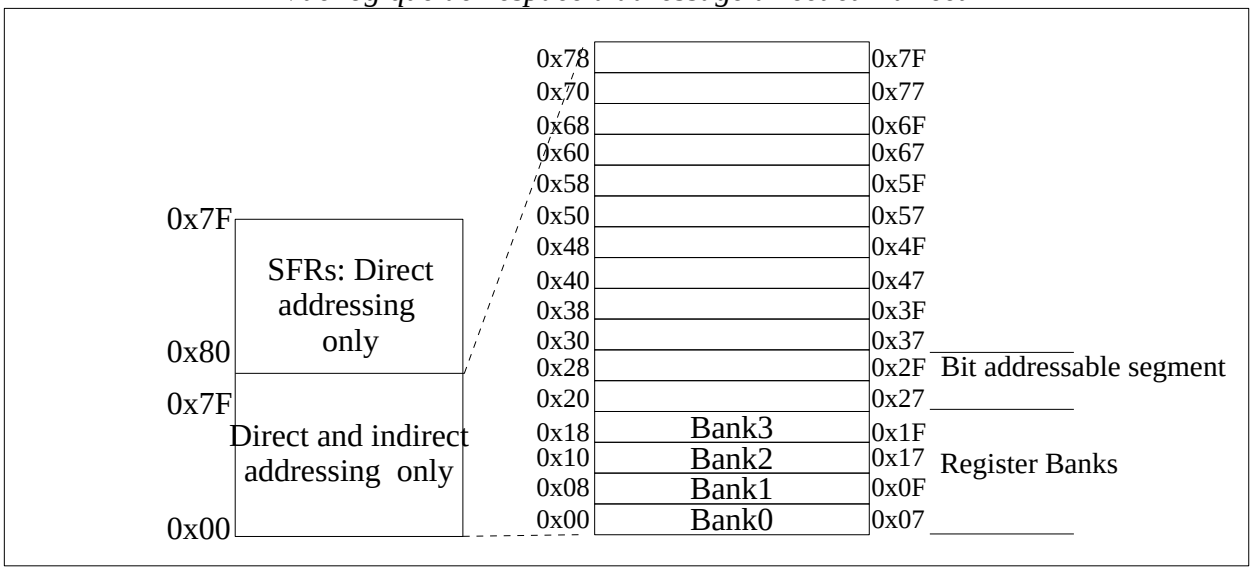
CY	PSW.7	Carry Flag.
AC	PSW.6	Auxiliary Carry Flag.
F0	PSW.5	Flag 0 available to the user for general purpose.
RS1	PSW.4	Register Bank selector bit 1
RS0	PSW.3	Register Bank selector bit 0
OV	PSW.2	Overflow Flag.
F1	PSW.1	Usable as a general purpose flag.
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of ‘1’ bus in the accumulator.

- A: Accumulator (8 bits)
- B: B register (8 bits)
- PC: Program counter (16 bits)
- DPTR: Data pointer (16 bits), reachable through the SFR interface (mapped to DPL, DPH)
- SP: Stack pointer (8 bits)

2.2.4 Espaces d'adressage

- Mémoire interne: Pile interne, implémentée en banc de latchs ou registres.

12 Vue logique de l'espace d'adressage direct et indirect



13 Le « mapping » SFR

0xF8		IE0	IE1	ITCTLCON					0xFF
0xF0	B	IPH0	IPH1						0xF7
0xE8		IPL0	IPL1						0xEF
0xE0	ACC								0xE7
0xD8									0xDF
0xD0	PSW								0xD7
0xC8									0xCF
0xC0									0xC7
0xB8									0xBF
0xB0									0xB7
0xA8									0xAF
0xA0									0xA7
0x98									0x9F
0x90									0x97
0x88								TSTAT	0x8F
0x80		SP	DPL	DPH	DOUT0	TCTRL	CNTH	CNTL	0x87

La colonne grisée correspond aux registres SFRs accessibles bit à bit.

Les SFRs sont des registres physiquement instanciés dans les interfaces des périphériques. Ils sont mappés logiquement dans l'espace d'adressage direct.

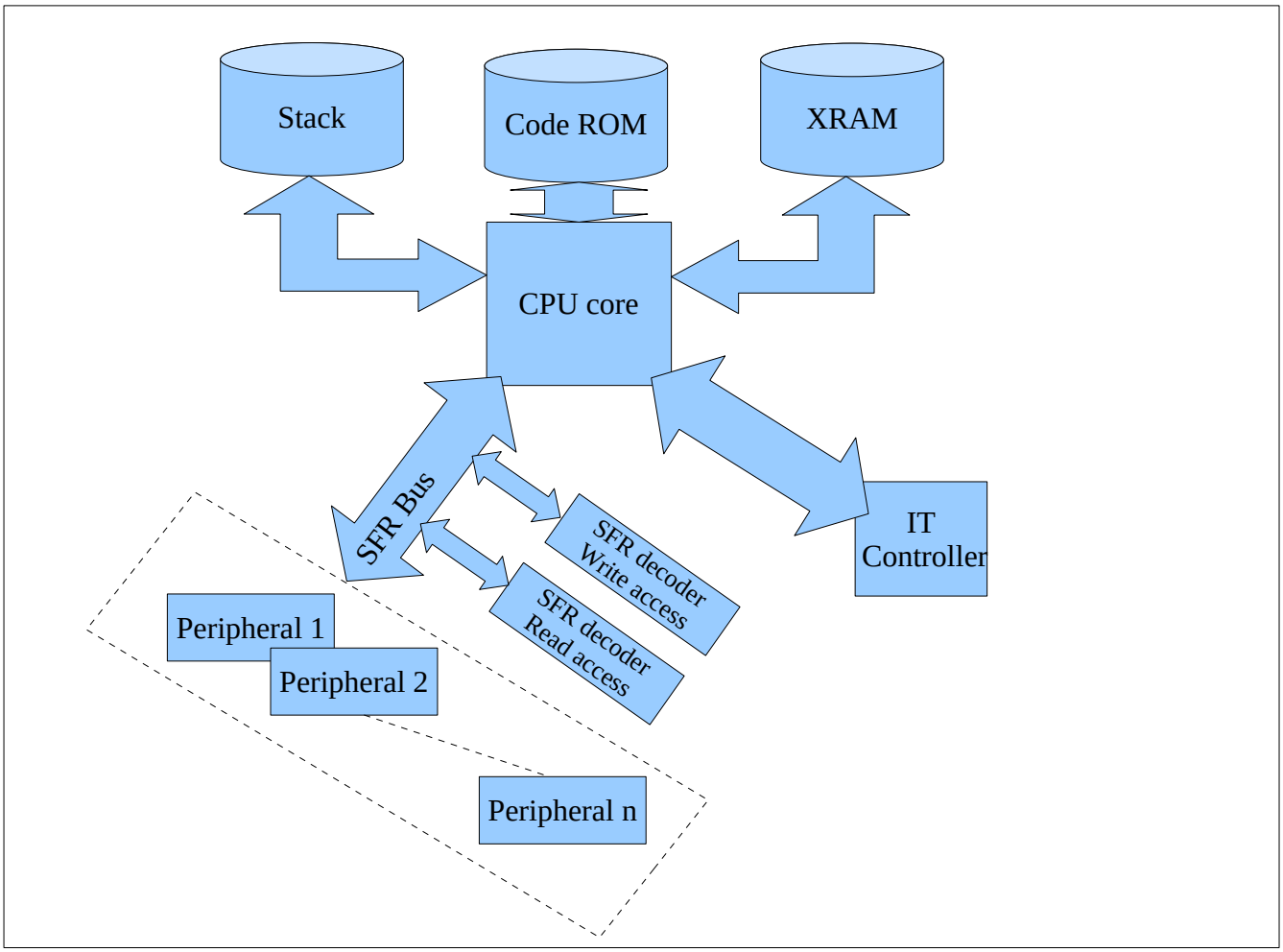
- Mémoire externe: 64K mots de 8bits accessibles en lecture et écriture via les instructions « movx » (cf. jeu d'instructions en annexe)
- Espace code:

  - accès utilisateur: 64K mots de 8bits accessibles en lecture seule via les instructions « movc » (cf. jeu d'instructions en annexe)

  - accès système: Le 51 execute toujours le code à l'adresse 0x0000 après reset. A partir de l'adresse 0x0003, se trouvent les adresses de déroutements d'interruptions. Soit i l'indice de l'IT traitée, le CPU sautera à l'adresse 3+i*8 (en décimal).

2.3 Architecture système

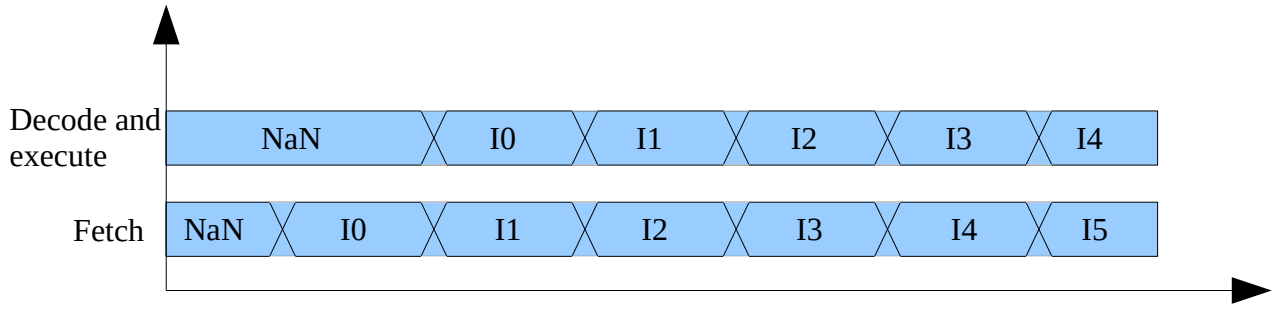
14 Architecture système



Un périphérique a un cœur et une interface, le bus SFR permet d'accéder en lecture et écriture aux registres instanciés dans l'interface des périphériques.

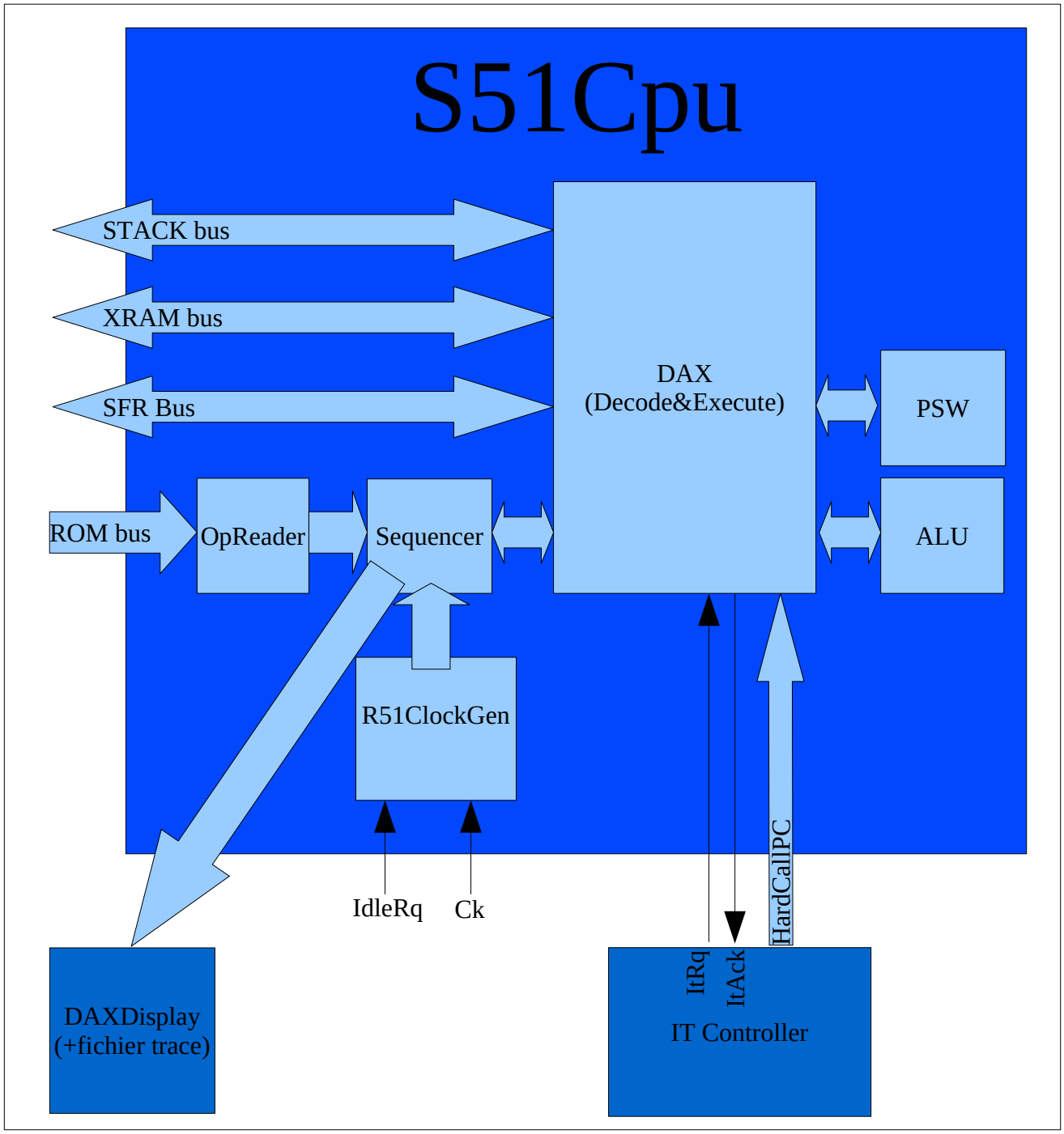
Le S51 a deux niveaux de pipeline:

- Fetch : Lecture des instructions
- Decode and execute : Décodage des instructions et exécution



2.4 Architecture du cœur S51

15 Architecture du cœur S51



Le bloc logique DAX contrôle les MUX qui sélectionnent les données à présenter sur les bus ainsi que leurs signaux de contrôle (read/write).

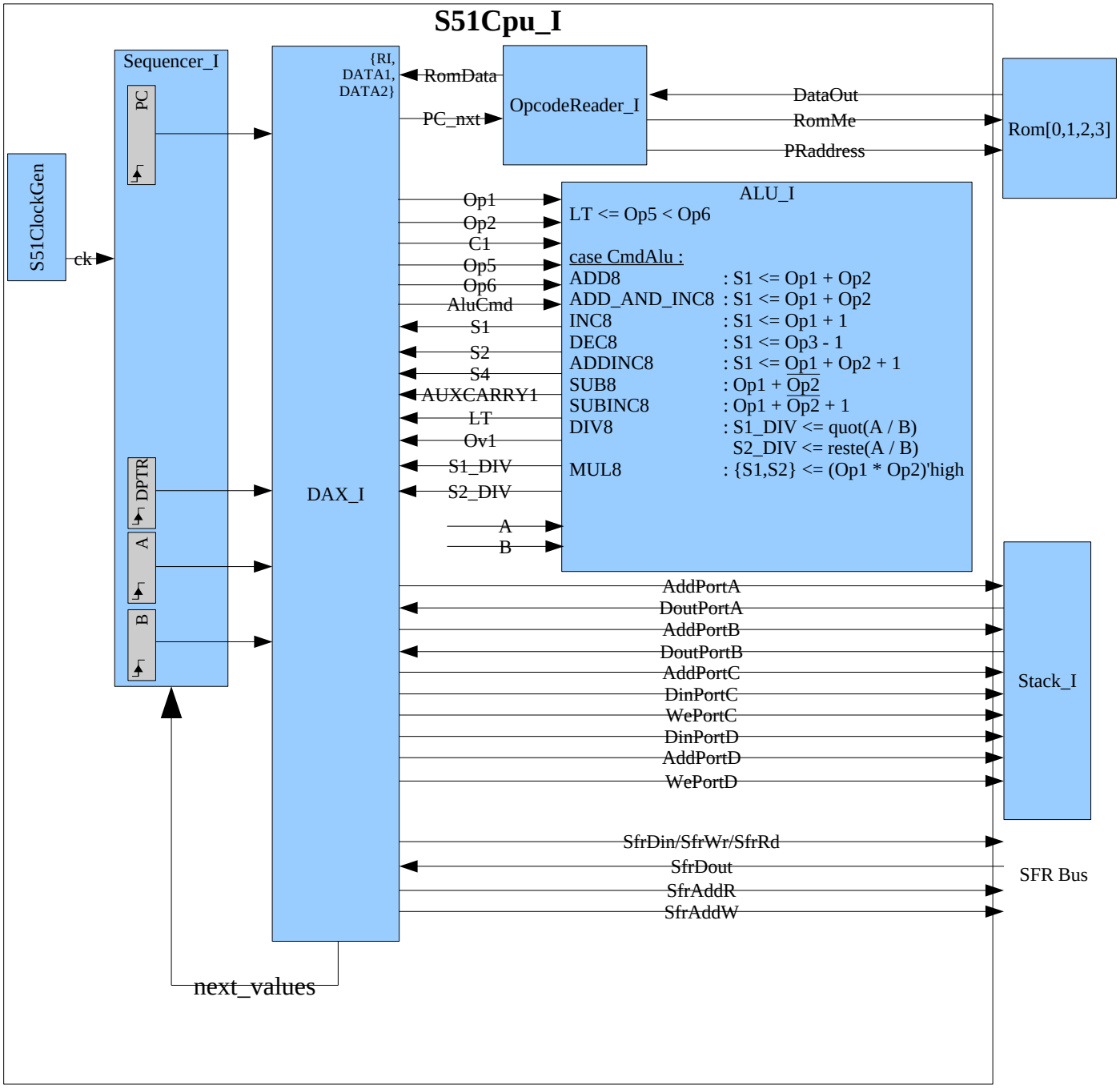
2.5 Implémentation du jeu d'instructions

La technique d'implémentation du jeu d'instructions utilisée pour le S51 est le « câblage » à opposer à la technique du « micro-codage », procédé très répandu dans les architectures RISC modernes car il permet d'obtenir les vitesses les plus élevées.

Nom du contrôle	Fonction (les commandes sont extraites du fichier PkgDAX.vhdl)
OpNb	Taille en octet de l'instruction (valeurs possibles: 1, 2, 3)
PMSelOffsetSrc	Sélectionne un offset à ajouter au PC dans le cas de sauts conditionnels réalisés (commandes: PMSELOFFSETNONE, PMSELOFFSETDATA1, PMSELOFFSETDATA2)
PMCONDSEL	Sélectionne la condition pour valider l'ajout d'un offset au PC utilisé dans le cas de sauts conditionnels (commandes: PMC_NULL, PMC_NOTACCUEQDIRECTDOUT, PMC_NOTACCUEQDATA1, PMC_NOTDOUTPORTAEQDATA1, PMC_NOTDOUTPORTBEQDATA1, PMC_DECDIRECTDOUTNOTNULL, PMC_DECDOUTPORTANOTNULL, PMC_TMPDAMBIT, PMC_PSWREGCY, PMC_NOTTMPDAMBIT, PMC_NOTCARRY, PMC_NOTACCUNULL, PMC_ACCUNULL, PMC_VCC)
CmdStack	Opération à mener sur la pile (commandes: STACK_NOOP, ReadRn, DecRn, XchRn, ReadIndRi, DecIndRi, StoreAccuToIndRi, StoreDataToIndRi, XchIndRi, XchdIndRi, Load8, Load16, StoreDirectToIndirectRi, StoreDirectToRn, StoreDataToRn, StoreAccuToRn, Store8, Store16, ReadRi)
CmdPCiMUX	Sélectionne la source de donnée à positionner pour la future valeur de PC (commandes: PM_NOOP, PM_CALL, PM_LCALL, PM_RET, PM_JMP, PM_JMPATAPLUSDPTR, PM_JMPATAPLUSPC)
CmdDam	Commande pour accès directs (impacte les bus SFR et Stack) (commandes: DAM_NOOP, DIRECT_LOAD_PORTA, DIRECT_READ_MODIFY_WRITE_PORTAC, DIRECT_STORE_PORTC, DIRECT_COPY, ClrBit, SetBit, CplBit, ReadBit, StoreCarryToBit, WriteDirectDintoIndRi, ReadSfrIndRi)
CMDOP1SEL	Sélectionne le bus d'entrée 1 de l'ALU (commandes: OP1_NONE, OP1ACCU, OP1DIRDOUT, OP1DOUTPORTA, OP1DOUTPORTB, OP1STACKPOINTER)
CMDOP2SEL	Sélectionne le bus d'entrée 2 de l'ALU (commandes: OP2_NONE, OP2DIRDOUT, OP2BREG, OP2DOUTPORTA, OP2DOUTPORTB, OP2DATA1, OP2TWO, OP2TWOHUNDREDFIFTYFOUR, OP2SIXTY, OP2SIX, OP2SIXTYSIX)
CMDACCUISEL	Sélectionne la source de donnée à positionner pour la future valeur du registre ACCU (commandes: ACCUI_NONE, ACCUIS1, ACCUIEQDIRECTDOUT, ACCUIEQDOUTPORTA, ACCUIEQDOUTPORTB, ACCUIEQACCUANDDIRECTDOUT, ACCUIEQACCUANDDOUTPORTA, ACCUIEQACCUANDDATA1, ACCUIEQACCUORDIRECTDOUT, ACCUIEQACCUORDOUTPORTA, ACCUIEQACCUORDOUTPORTB, ACCUIEQACCUORDATA1, ACCUIEQACCUXORDIRECTDOUT, ACCUIEQACCUXORDOUTPORTA, ACCUIEQACCUXORDATA1, ACCUIEQACCUXORDOUTPORTB, ACCUIRL, ACCUIRLC, ACCUIRR, ACCUIRRC, ACCUISWAP, ACCUIXCH, ACCUIEQCPL, ACCUIEQNULL, ACCUIEQMOVCDATA, ACCUIEQXRAMDOUT, ACCUIEQACCUANDDOUTPORTB, ACCUIEQDATA1)
CMDDIRECTDINSEL	Sélectionne la source de donnée à positionner sur le bus SfrDin (commandes: DIRECTDIN_NONE, DIRECTDINEQDIRECTDOUTANDACCU, DIRECTDINEQDIRECTDOUTORACCU, DIRECTDINEQDIRECTDOUTORDATA2, DIRECTDINEQDIRECTDOUTXORDATA2, DIRECTDINEQDIRECTDOUTANDDATA2, DIRECTDINEQDIRECTDOUTXORACCU, DIRECTDINS1, DIRECTDINACCU,

	DIRECTDINDECDIRECTDOUT, DIRECTDINDIRECTDOUT, DIRECTDINDOUTPORTA, DIRECTDINDOUTPORTB, DIRECTDINDATA2)
CMDOP3SEL	Sélectionne le bus d'entrée 3 de l'ALU (commandes: OP3_NONE, OP3ACCU, OP3DIRECTDOUT, OP3DOUTPORTA, OP3DOUTPORTB, OP3STACKPOINTER)
CMDSTACKDINSEL	Sélectionne une des sources possibles pour le bus DIN du port C de la pile (commandes: STACKDIN_NONE, STACKDINS1, STACKDINDECDOUTPORTA)
CMDSTACKPOINTERSEL	Sélectionne la source du registre SP (Soit SP lui même soit la sortie S1 de l'ALU) (commandes: CMDSTACKPOINTER_NONE, CMDSTACKPOINTERS1)
CMDOP5SEL	Sélectionne le bus d'entrée 5 de l'ALU (commandes: OP5ACCU_NONE, OP5ACCU, OP5DOUTPORTA, OP5DOUTPORTB)
CMDOP6SEL	Sélectionne le bus d'entrée 6 de l'ALU (commandes: OP6_NONE, OP6DIRECTDOUT, OP6DATA1)
CMDDPTRISEL	Sélectionne la source de donnée à positionner pour la future valeur du registre DPTR (commandes: CMDDPTRI_NONE, CMDDPTRIINC, CMDDPTRIDATA16)
CMDXRAMDINSEL	Sélectionne la source de donnée à positionner sur le bus XRAMDin (commandes: CMDXRAMDIN_NONE, CMDXRAMDINEQACCU, CMDDPTRIDATA16)
CMDXRAMADDRESS	Sélectionne la source de donnée à positionner sur le bus XRAMAdd (commandes: CMDXRDPTR, CMDXRDOUTPORTA)
CMDBREGISEL	Sélectionne la source de donnée à positionner pour la future valeur du registre B (commandes: BREGI_NONE, BREGIS2)

2.6 Architecture du cœur S51 (vue détaillée)



2.7 Exemple de décodage d'instruction:

2.7.1 Cas du décodage de l'instruction : MOV DPTR, #DATA16

Extrait datasheet:

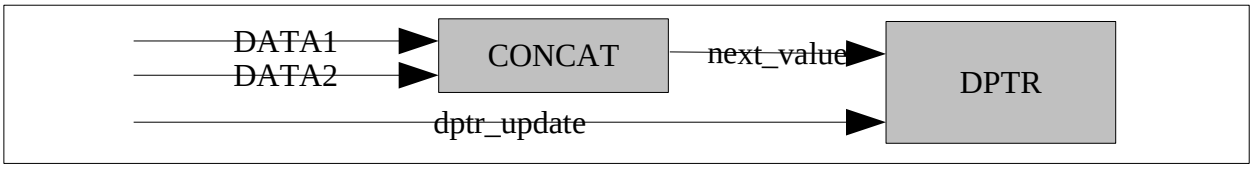
MOV DPTR,#data16
Function: Load Data Pointer with a 16-bit constant

Description: The Data Pointer is loaded with the 16-bit constant indicated. The 16-bit constant is loaded into the second and third bytes of the instruction. The second byte (DPH) is the high-order byte, while the third byte (DPL) holds the low-order byte. No flags are affected.
This is the only instruction which moves 16 bits of data at once.
Example: The instruction,
MOV DPTR,#1234H
will load the value 1234H into the Data Pointer: DPH will hold 12H and DPL will hold 34H.
Bytes: 3
Encoding:

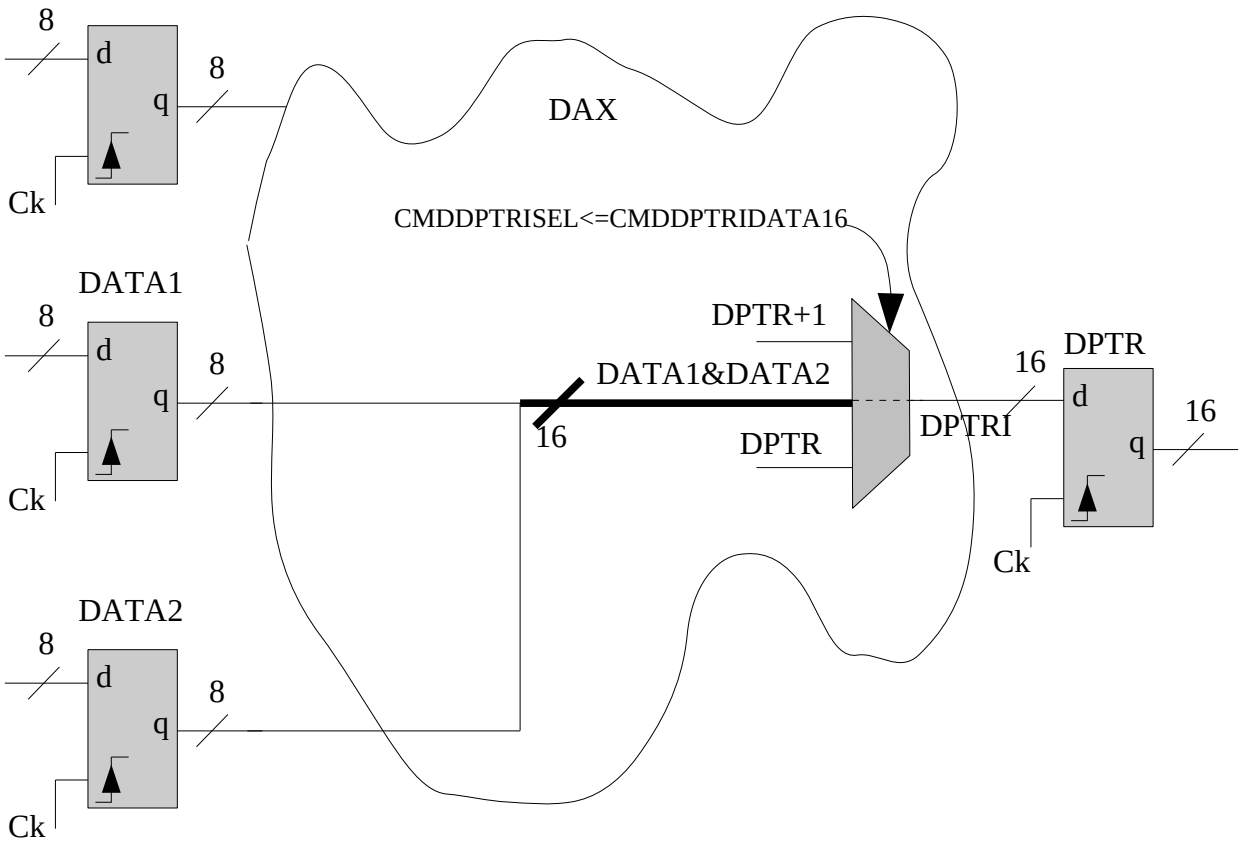
1 0 0 1 0 0 0 0	immed. data15-8	immed. data7-0

Operation: MOV
(DPTR) <= (#data$_{15-0}$)
DPH&DPL <= #data$_{15-8}$ #data$_{7-0}$
Cycles: 1

Vue fonctionnelle:



RI= « MOV DPTR, #DATA16 »



2.7.2 Cas du décodage de l'instruction : MOV DIR, DIR

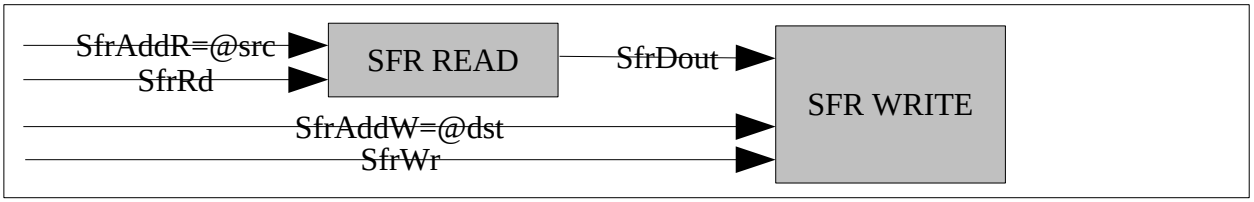
Extrait datasheet:

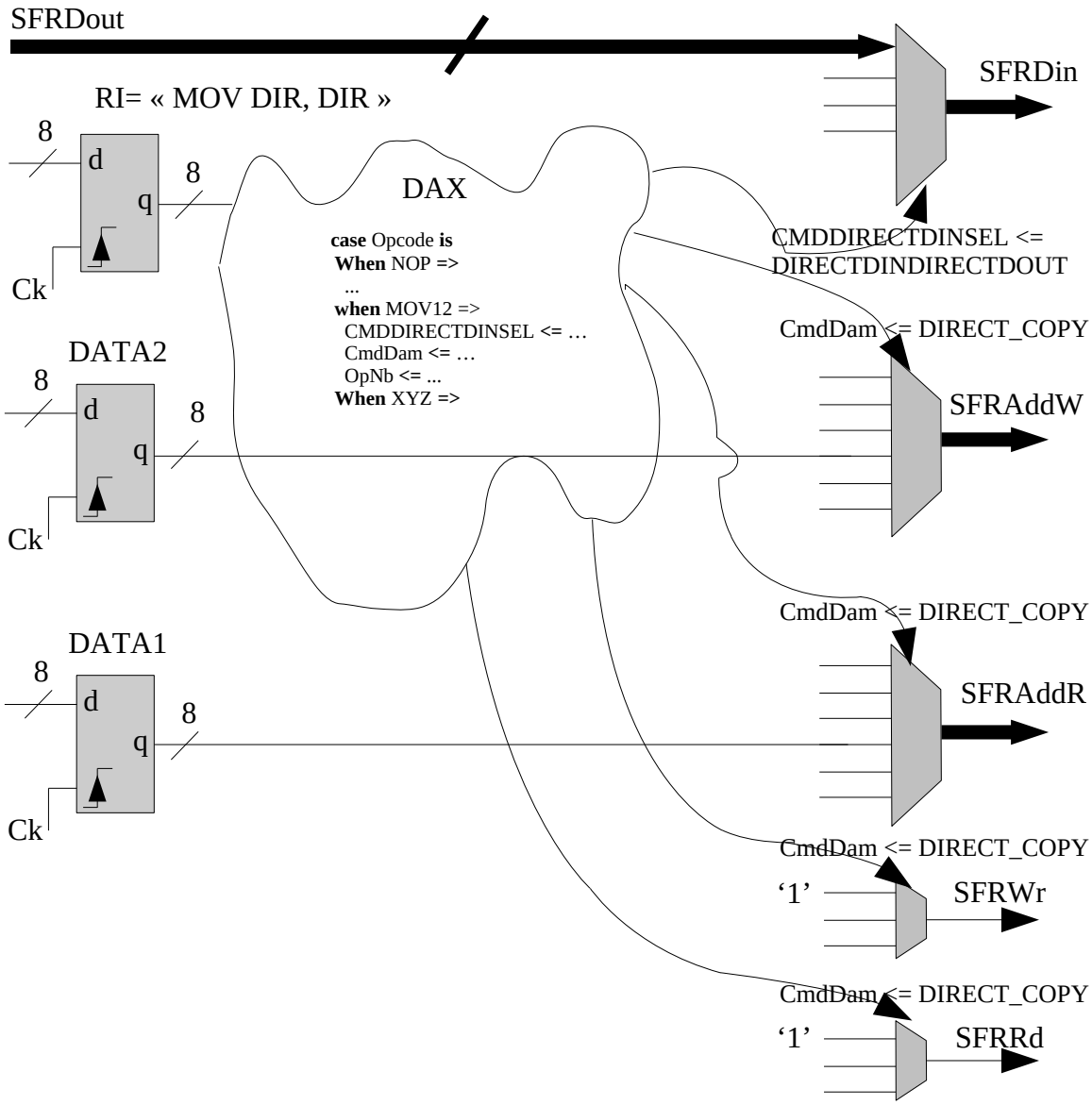
MOV direct,direct
Bytes: 3
Encoding:

1 0 0 0 0 1 0 1	dir. addr. (src)	dir. addr. (dest)

Operation: MOV
(direct) <= (direct)
Cycles: 1

Vue fonctionnelle:





Un chronogramme illustrant les accès type sur le bus SFR est représenté dans la section 2.10.2 « Bus Périphérique (SFR)» page 33.

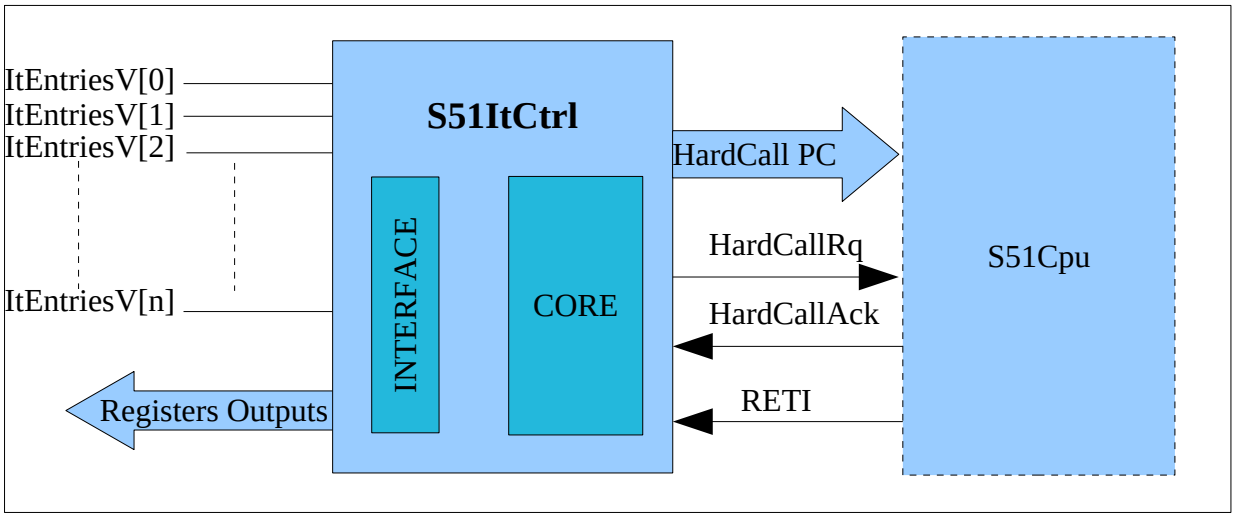
2.8 Les interruptions

Les interruptions gérées par le S51 le sont sur niveau (à opposer à un fonctionnement sur événement), c'est à dire qu'un signal d'interruption doit rester actif tant que l'interruption n'a pas été traitée. Lorsqu'un périphérique génère une interruption, le CPU est dérouté sur la routine correspondante et doit éteindre la source d'interruption à l'aide d'une écriture SFR dans un registre de contrôle du périphérique.

Le contrôleur d'interruptions gère jusqu'à 16 entrées, à chacune est associée un poids codé sur 2 bits ainsi qu'un bit d'enable. Ainsi à un instant « t », il peut y avoir 4 routines d'interruptions en cours d'exécution sachant qu'une routine d'interruption n'est interruptible que par une interruption de priorité strictement supérieure à la sienne.

2.8.1 Schéma bloc

16 Schéma bloc



2.8.2 Interface utilisateur

- registres de configuration

Registre ITCTLCON: registre de configuration

Bit number	7	6	5	4	3	2	1	0
Field name	-	-	-	-	-	-	-	EN
Access	-	-	-	-	-	-	-	RW
Rst value	-	-	-	-	-	-	-	0

Registre IE0: configuration des 8 lignes d'interruption actives de poids faible

Bit number	7	6	5	4	3	2	1	0
Field name	L7	L6	L5	L4	L3	L2	L1	L0
Access	RW	RW	RW	RW	RW	RW	RW	RW
Rst value	0	0	0	0	0	0	0	0

Registre IE1: configuration des 8 lignes d'interruption actives de poids fort

Bit number	7	6	5	4	3	2	1	0

Field name	L15	L14	L13	L12	L11	L10	L9	L8
Access	RW	RW	RW	RW	RW	RW	RW	RW
Rst value	0	0	0	0	0	0	0	0

Registre IPH0: configuration du bit de poids fort des 8 lignes d'interruption de poids faible

Bit number	7	6	5	4	3	2	1	0
Field name	WH7	WH6	WH5	WH4	WH3	WH2	WH1	WH0
Access	RW	RW	RW	RW	RW	RW	RW	RW
Rst value	0	0	0	0	0	0	0	0

Registre IPL0: configuration du bit de poids faible des 8 lignes d'interruption de poids faible

Bit number	7	6	5	4	3	2	1	0
Field name	WL7	WL6	WL5	WL4	WL3	WL2	WL1	WL0
Access	RW	RW	RW	RW	RW	RW	RW	RW
Rst value	0	0	0	0	0	0	0	0

Registre IPH1: configuration du bit de poids fort des 8 lignes d'interruption de poids fort

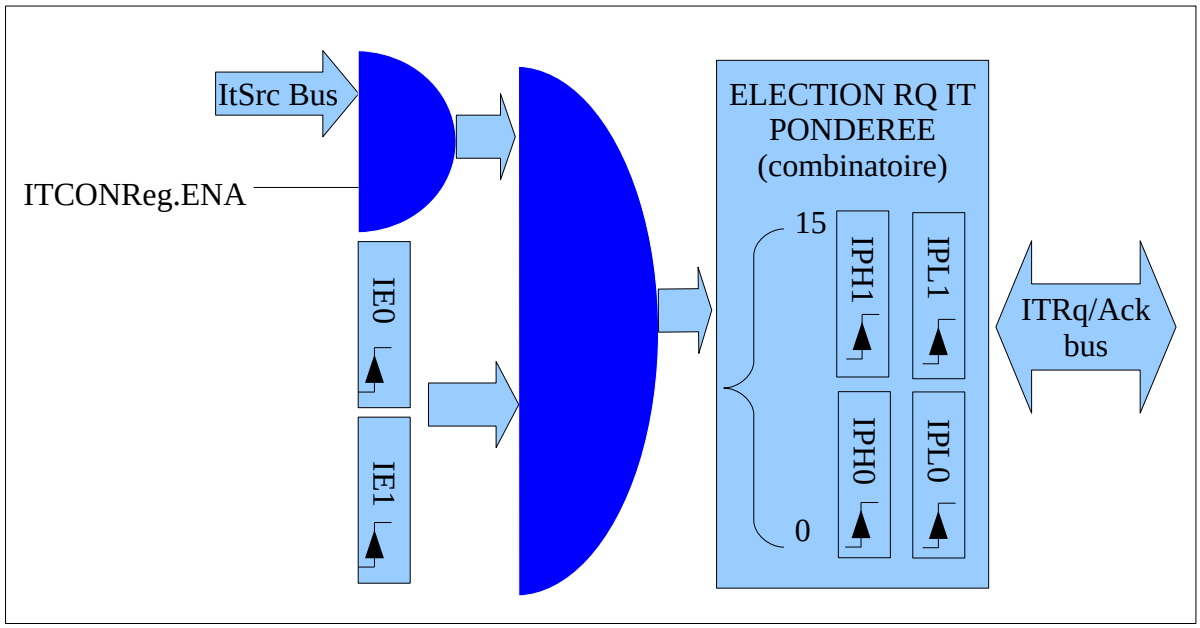
Bit number	7	6	5	4	3	2	1	0
Field name	WH15	WH14	WH13	WH12	WH11	WH10	WH9	WH8
Access	RW	RW	RW	RW	RW	RW	RW	RW
Rst value	0	0	0	0	0	0	0	0

Registre IPL1: configuration du bit de poids faible des 8 lignes d'interruption de poids fort

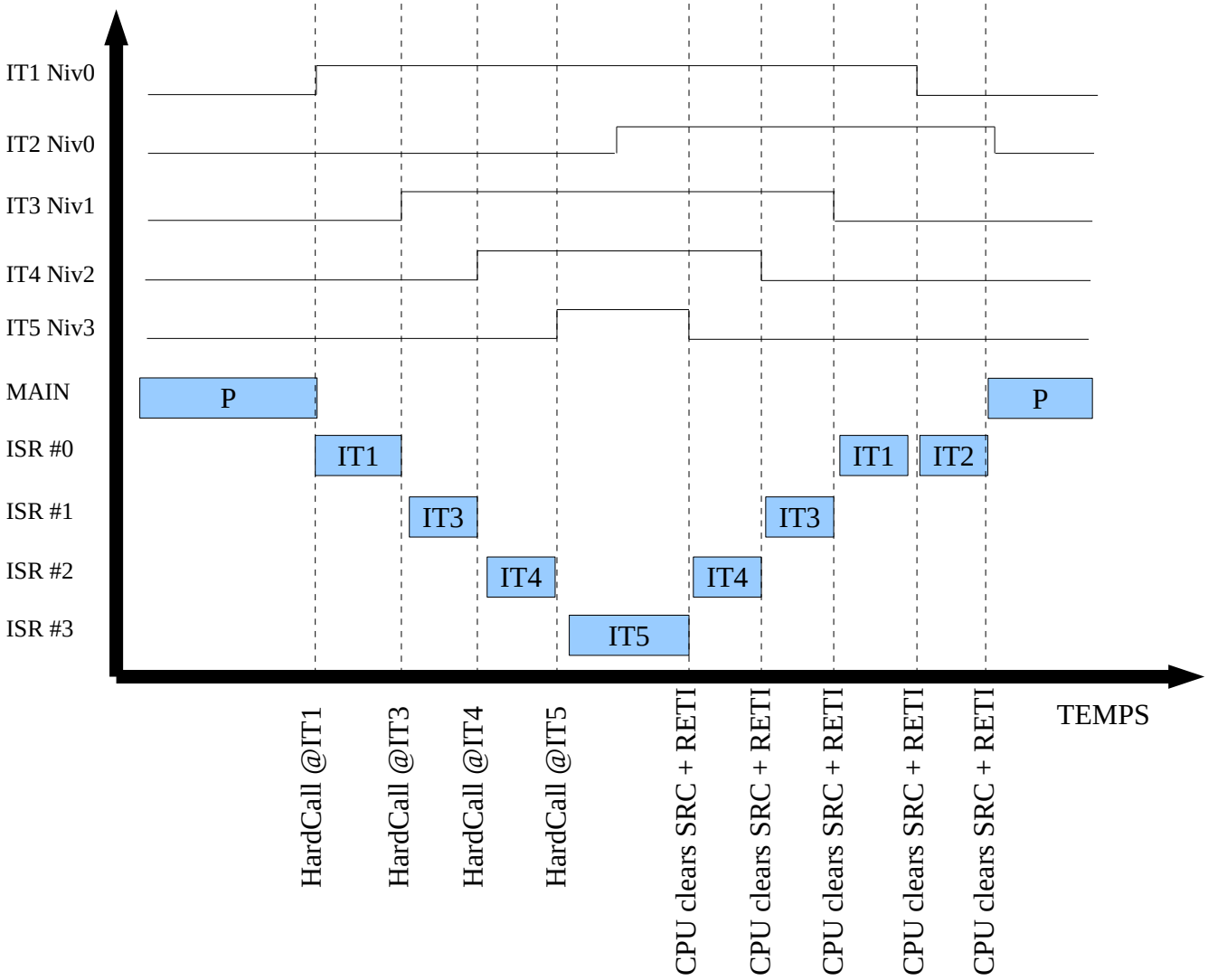
Bit number	7	6	5	4	3	2	1	0
Field name	WL15	WL14	WL13	WL12	WL11	WL10	WL9	WL8
Access	RW	RW	RW	RW	RW	RW	RW	RW
Rst value	0	0	0	0	0	0	0	0

2.8.3 Description fonctionnelle

17 Schéma de principe

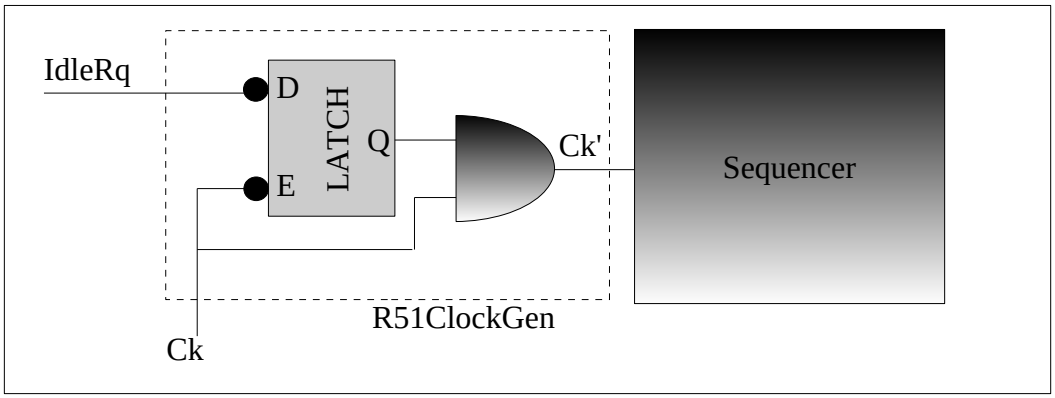


18 Transactions CPU/Contrôleur d'IT



2.9 Le mode IDLE

L'état du système dépend de l'état des bus d'entrée et des bascules du CPU. Le CPU étant maître sur l'ensemble de ces bus (excepté pour les interruptions), il suffit de geler l'ensemble des bascules du séquenceur du CPU. On fait du « clock gating » en utilisant une cellule spécifique à base de latch dont le schémas équivalent est présenté ci-dessous. Son principe est de bloquer le signal combinatoire de requête d'horloge sur la phase haute de la clock et de le laisser passer sur phase basse. Cela permet de laisser à ce signal une période complète pour s'établir puis de le bloquer dès la phase haute suivante de l'horloge pour filtrer les commutations pendant la période suivante. Sur la phase basse, ces commutations sont filtrées par le porte AND2.

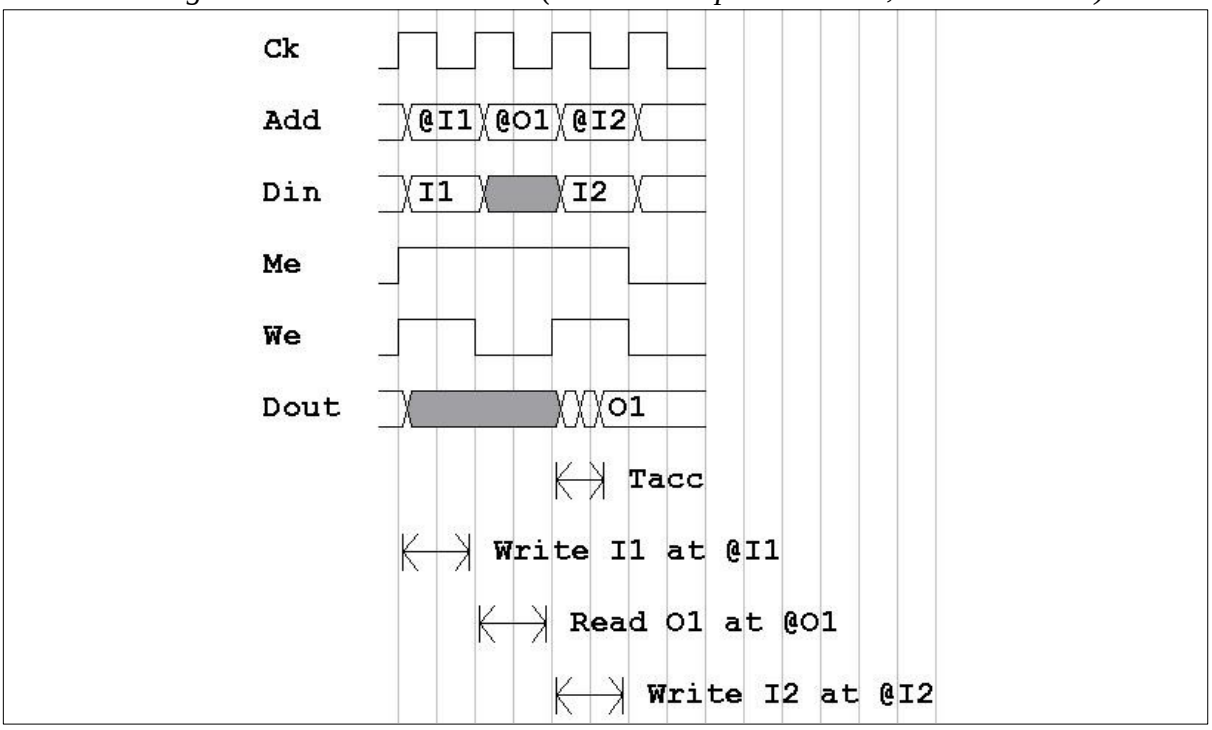


2.10 Description des bus

2.10.1 Bus mémoire (mémoire code et mémoire externe):

Les mémoires utilisées sont des mémoires synchrones. La mémoire code est dans le chemin critique du S51 (accès au code). Les accès sont réalisés sur le front montant de « Ck » si « M »e est à '1'. Si « We » est à '1' alors, c'est une lecture, sinon c'est une écriture.

19 Chronogramme des accès en RAM (même chose pour la ROM, sans Din et We)



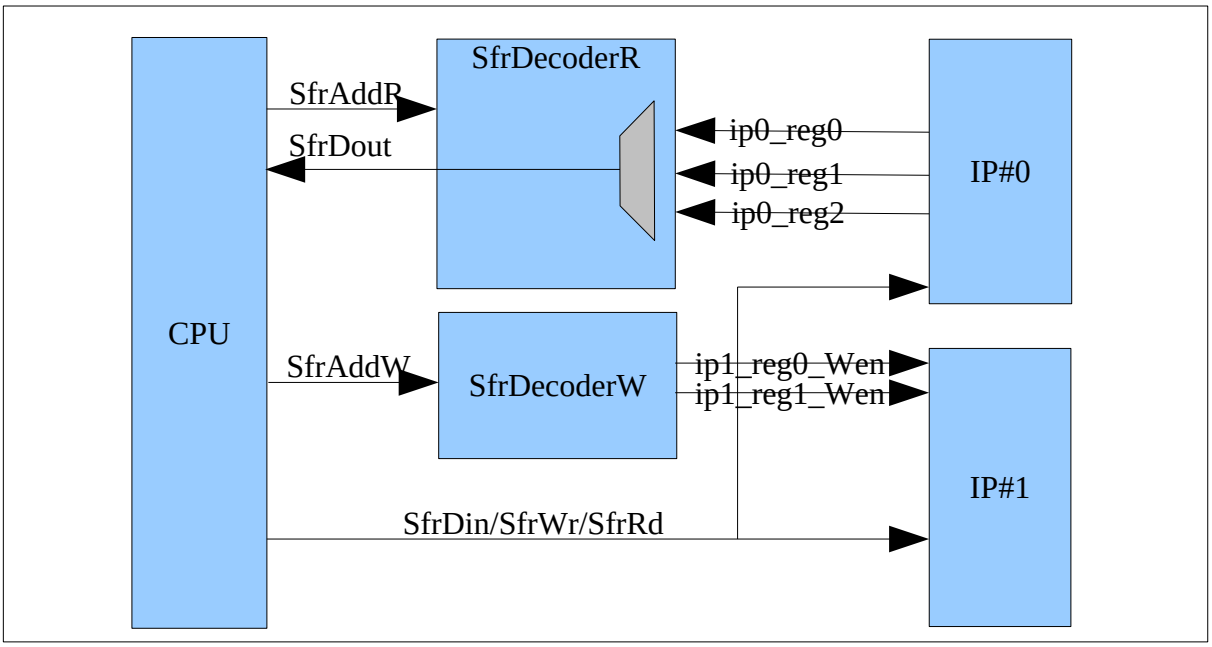
2.10.2 Bus Périphérique (SFR)

Le bus SFR est constitué de:

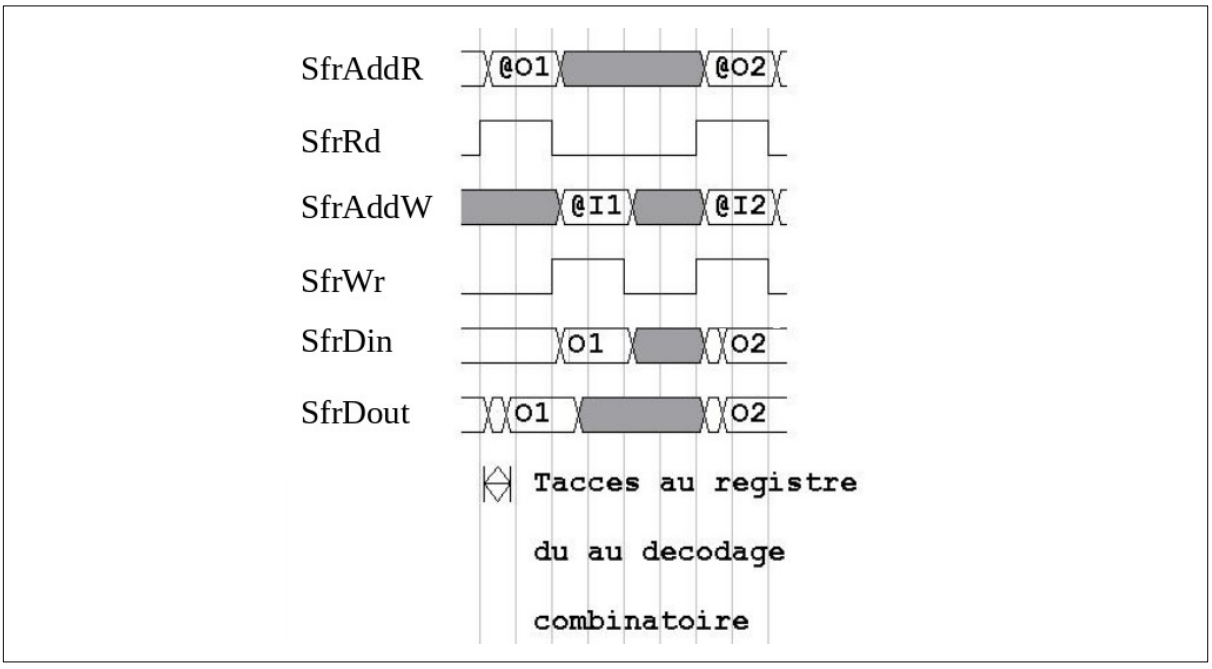
- 2 sous-bus d'adresse de 7 bits (1 bus pour accès en lecture, 1 bus pour accès en écriture)
- Commande de lecture SfrRd
- Commande d'écrite SfrWr

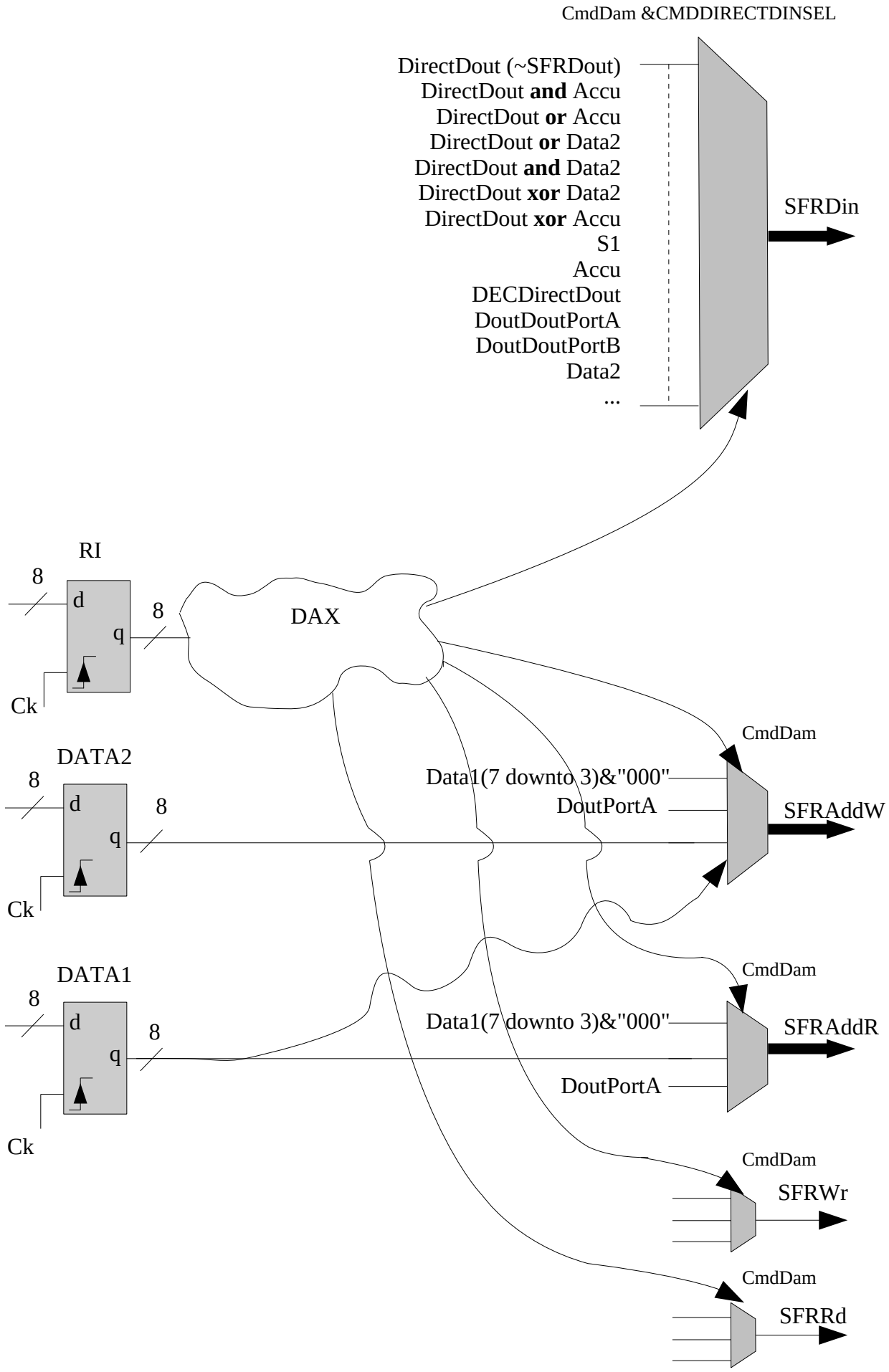
2 instances de « SfrDecoder » décodent chaque bus d'adresse (SfrAddR et SfdAddW) et sélectionnent les registres à accéder. A un instant donné, une et une seule écriture peut avoir lieu et simultanément une et une seule lecture peut avoir lieu. L'accès en lecture est un accès combinatoire (Taccès est le temps de traversée de la logique de décodage+temps de traversée des étages de MUX), l'écriture est synchrone.

20 Schéma bloc



21 Chronogramme des accès sur bus SFR




CmdDam &CMDDIRECTDINSEL
DirectDout (~SFRDout)
DirectDout **and** Accu
DirectDout **or** Accu
DirectDout **or** Data2
DirectDout **and** Data2
DirectDout **xor** Data2
DirectDout **xor** Accu
S1
Accu
DECDirectDout
DoutDoutPortA
DoutDoutPortB
Data2
...
SFRDin
RI
8
d
q
8
Ck
DAX
DATA2
8
d
q
8
Ck
CmdDam
Data1(7 downto 3)&"000"
DoutPortA
SFRAddW
DATA1
8
d
q
8
Ck
CmdDam
Data1(7 downto 3)&"000"
DoutPortA
SFRAddR
CmdDam
SFRWr
CmdDam
SFRRd

2.11 Testbench

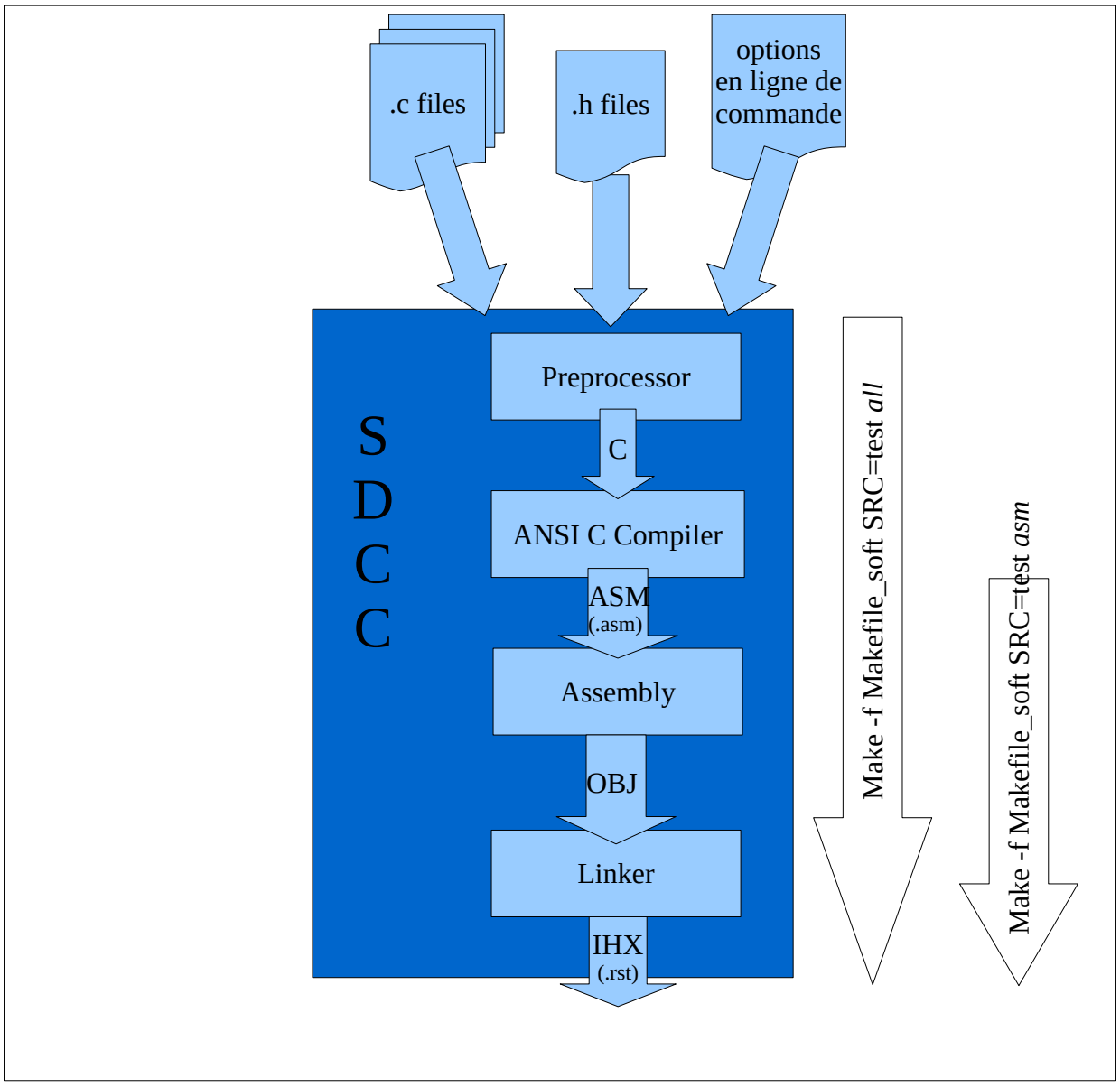
Il instancie le TopAppliS51 et lui fournit une horloge. Instancie également le moniteur système qui permet de tracer l'activité du CPU dans un fichier. Ce mécanisme permet de faire des non-régressions lors de la conception du CPU.

3 Outils de compilation C et ASM

3.1 SDCC

SDCC: Compilateur ANSI C sur cibles Intel 8051, Maxim 80DS390, Zilog Z80 et Motorola 68HC08.

22 SDDC: Small Device C Compiler



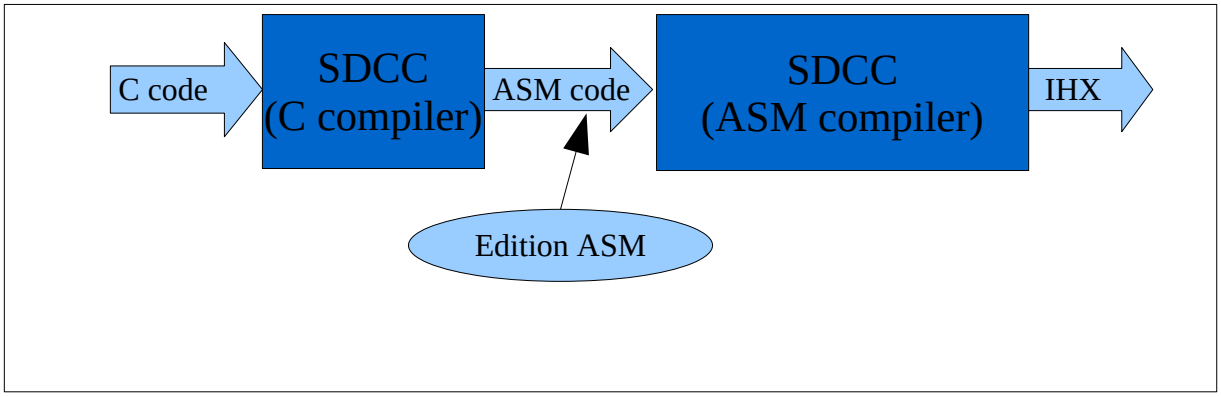
Le répertoire SCRIPTS contient le fichier Makefile_soft. Pour compiler un fichier C au format texte pour la ROM du circuit exécuter la commande suivante :

make -f Makefile_soft SRC= « NOM_DU_PATTERN_SANS EXTENSION » all

Un fichier de sortie rom0.txt est écrit qui contient le code à charger en ROM. Ce fichier sera chargé par le modèle de mémoire lors de la simulation.

3.2 Flow de compilation en deux étapes (patch assembleur)

23 Flow de compilation en deux étapes (cf. fichier source/SCRIPTS/Makefile_soft)



Makefile_soft gère la compilation du code C en deux phases :

- compilation ASM
- Assemble+link

ce qui permet de « patcher » le code assembleur.

4 Démarrer la synthèse logique

Synthèse du timer :

make synth_tim

Synthèse chip :

make synth_chip

L'outil de synthèse logique est «RTL compiler» de CADENCE. Les deux cibles démarrent l'interface graphique. Après chaque synthèse, il faut quitter l'interface graphique pour relâcher la licence (menu : file->exit).

5 Démarrer l'analyse statique de timing

On utilisera l'outil d'analyse statique de timing du synthétiseur. Si une synthèse a déjà été réalisée, on peut recharger la dernière lancée session avec la commande suivante :

make synth_chip_restore

Assurez vous qu'il n'y ait pas de latchs dans votre design.
Pour lancer le gui, utiliser la commande : *gui_show*

Pour avoir une vue d'ensemble des chemins longs : allez dans le menu « timing » puis « report » et choisir « Endpoint Histogram ».

La documentation des C35 standard cells est disponible ici :
*opt*AMC_37x/www/databooks/c35/databook_c35_33/index.html

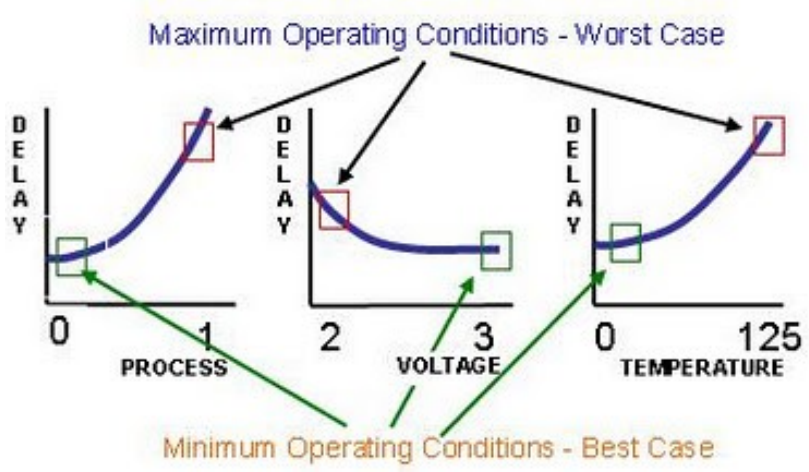
6 Analyse statique de timings

L'analyse statique de timing permet de vérifier que les caractéristiques du circuit vérifient les contraintes de timings spécifiées. Néanmoins il est nécessaire de fournir à l'outil des informations pour ne pas sur-contraindre la synthèse. On peut donc spécifier par exemple des faux chemins ou des chemins pour lesquels la logique dispose de plusieurs cycles pour s'établir. Référez vous au fichier de contraintes *IpS51_exceptions.tcl* dans le répertoire SCRIPTS.

Pour avoir la vue statique des sous-blocs du S51Cpu et leurs inter-connections, référez vous au schéma bloc de la section 2.6 « Architecture du cœur S51 (vue détaillée) ».

L'analyse statique de timing permet également d'identifier des chemins qui pourraient être « relaxés » si fonctionnellement il est acceptable d'augmenter leur temps d'établissement. Par exemple, l'instruction DIV est rarement utilisée pour une grande majorité d'applications. Dans ce cas, on peut modifier le design pour que cette instruction prenne plusieurs cycles plutôt qu'un seul. Si on laisse fonctionnellement plusieurs cycles au chemin le plus long pour s'établir, cela permet d'augmenter la fréquence du circuit. L'ensemble des autres instructions s'exécutera plus rapidement. Globalement, on améliore les performances du circuit. L'idée est d'équilibrer la complexité des chemins.

6.1 PVT operating conditions



- Best design corner (Best-case) : fast process, highest voltage and lowest temperature
- Worst design corner (Worst-case) : slow process, lowest voltage and highest temperature

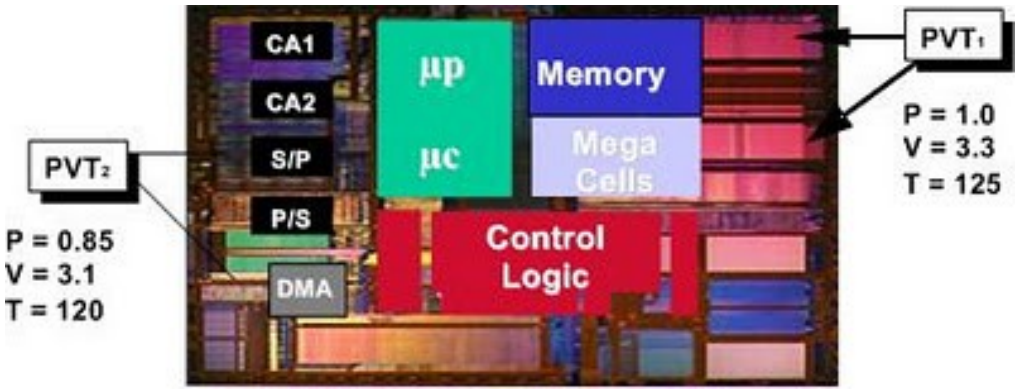
6.2 « On-chip » variation timing analysis

L'analyse « on-chip variation » prend en compte des différences de timings mineures sur différentes parties de la puce dans un corner (pvt) donné. Dans cette analyse les chemins « early data » et « clock » sont analysés dans les conditions min et les chemins « late data » et « clock » dans les conditions max en appliquant un facteur « k » de dégradation de timing.

Ces variations sur une même puce (ou un « die ») sont dues aux:

- Variations des procédés de fabrication (P)
- Variations de tension dues à l'activité du circuit (IR drop)
- Variations de température (hot-spots ...)

On modélise ces variations à l'aide de coefficients de dégradation que l'on donne à l'outil et proviennent d'analyses de caractérisation des librairies.



7 Démarrer la simulation RTL

7.1 Compilation de la base de simulation avec modelsim

Tous les fichiers source VHDL nécessaires sont dans le répertoire VHDL. Pour les compiler exécutez la commande suivante :

cd SCRIPTS

make compile_msim

Les librairies compilées sont générées dans le répertoire WORK.

7.2 Compilation d'une application de test

Pour créer le fichier texte qui servira à initialiser le plan mémoire du modèle comportemental de la ROM exécutez la commande suivante (dans le répertoire SCRIPTS):

make -f Makefile_soft SRC=test all

modifiez le fichier assembleur résultant pour sauter la routine d'initialisation des variables et plans mémoires qui fait appel à des instructions non implémentées:

gedit test.asm

puis ligne 177, remplacez:

__interrupt_vect: *ljmp __sdcc_gsinit_startup*	par	__interrupt_vect: ljmp _main

7.3 Simulation avec modelsim

Dans une console UNIX, sous le répertoire SCRIPTS :

make sim

ou alors dans le répertoire WORK, lancez la commande vsim

- Vous pouvez charger le design via l'interface graphique
- Pour gagner du temps vous pouvez charger le design en ligne de commande comme suit :

vsim -novopt -t ps +nospecify LibTestBench.testbench

note :

-novopt disables optimisations causing some signals to be remove from the simulation database

+nospecify disables specify path delays and timing checks

-t ps : Time resolution limit

Dans la console *modelsim*, chargez les « waveform » des signaux élémentaires avec la commande suivante : ***do wave.do***

Observez le vecteur « *accu* ».

Note : Le code qui sera exécuté sera celui compilé la dernière fois dans le répertoire *SCRIPTS*.

8 Démarrer la simulation NTL

8.1 Compilation avec modelsim

La compilation de la netlist se fait avec les commandes suivantes :

cd SCRIPTS

make compile_msim_synthams

Pour ne recompiler que le testbench après changement de la fréquence d'horloge :

make compile_msim_synthams_tb

Les librairies *modelsim* compilées sont générées dans le répertoire WORK_NTL.

8.2 Simulation sans timings avec modelsim

Dans une console UNIX, positionnez vous dans le répertoire SCRIPTS.

Puis exécutez:

make sim_ntl

note à propos des options vsim utilisées dans la target sim_ntl :

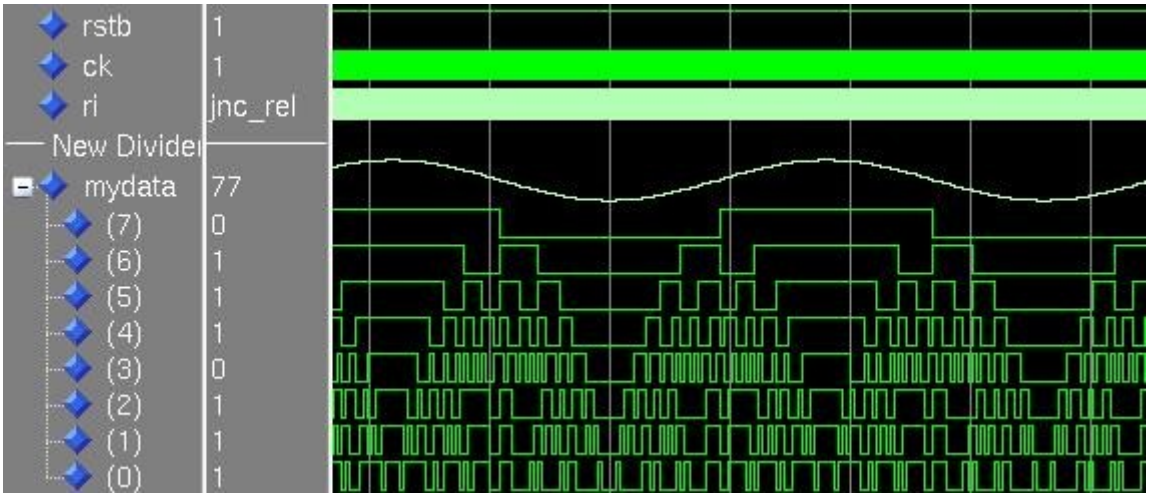
-novopt disables optimisations causing some signals to be remove from the simulation database

+nospecify disables specify path delays and timing checks

-t ps : Time resolution limit

Dans la console *modelsim*, chargez les « waveform » des signaux élémentaires avec la commande suivante : ***do wave.do***
exemple avec le pattern gensin_poll.c :

24 Simu sans SDF (la forme sinusoïdale est un mode de représentation du vecteur numérique)



8.3 Simulation rétro-annotée avec modelsim

Même chose que dans la section précédente.

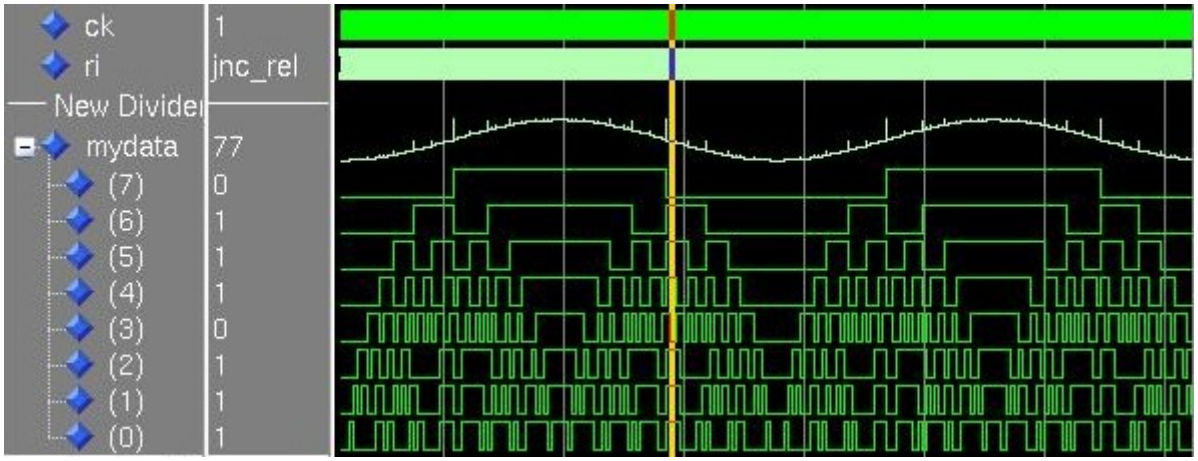
La commande à exécuter pour charger le SDF est:

make sim_synth

exemple avec le pattern gensin_poll.c :

Bruit constaté après rétro-annotation dû au fait que les sorties des 8 DFFs du vecteur « Dout0 » ne commutent pas en même temps.

25 Simu avec SDF (la forme sinusoïdale est un mode de représentation du vecteur numérique)



Note : Le code qui sera exécuté sera celui compilé la dernière fois dans le répertoire *SCRIPTS*.

9 Démarrer la simulation mixte

9.1 ADvance MS

ADMS encapsule un simulateur digital et un solver analog. Chaque simulateur prend en charge les modules de leur domaine respectif et *ADMS* fait communiquer les deux mondes à l'aide de convertisseurs qu'il insère automatiquement à l'élaboration de la base de simulation.

- Dans le cas d'une interface entre un modèle digital et un modèle *VHDL-AMS* ou *VERILOG-AMS*, *ADMS* utilise les informations liées aux terminaux qui ont une « nature » qui elle même obéit à une « discipline ».
- Dans le cas d'une interface entre un modèle digital et un modèle *SPICE*, *ADMS* utilise un fichier d'association « *.assoc* » qui détermine les interfaces entre les instances de modèles analogiques et digitaux. ADMS détermine les convertisseurs à instancier en fonction de la déclaration du type de port (IN=>D2A, OUT=>A2D, INOUT=>D2A+A2D) déclarés dans les sources VHDL.

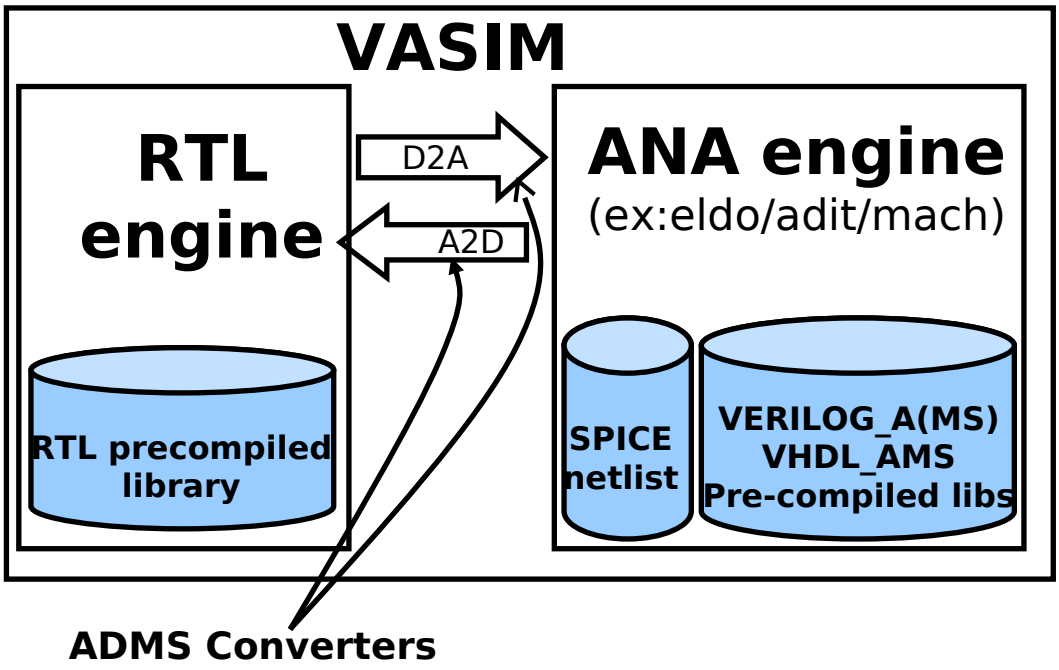
Exemple de fichier d'association pour modèle spice:

[Header]
Unit 1: Verilog WORK:nom_du_module_module
Unit 2: SPICE nom_du_subsck@chemin_d_acces_au_fichier_cir

[Port association] *<----- Associations de port entre domaines digital et analog*
a => a
b => b
c => c
[Unit 1 special] <---------- Association d'un port digital sur un nœud digital
d =>

[Unit 2 special] <---------- Association d'un terminal analog sur un nœud global analog
vdd => mon_pwl_vdd
gnd => mon_gnd

26 Schéma de principe du simulateur vasim d'ADMS



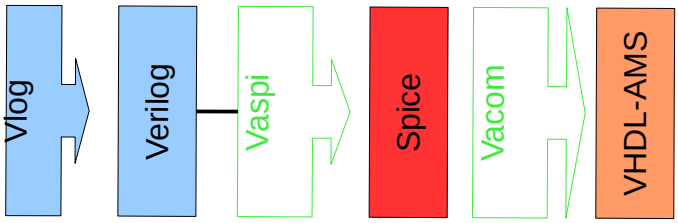
Dans la structure du projet:

- Le flow de compilation ADMS est décrit dans la section compile_vams du Makefile du répertoire SCRIPTS.
- Les modèles analogiques tels que ceux du BOD sont placés dans les répertoires TECH/SPICE, TECH/VHDLAMS, TECH/VHDL (cf. section 11.1 « Structure du répertoire projet» p°60.
- La simulation se lance dans le répertoire WORK_AMS

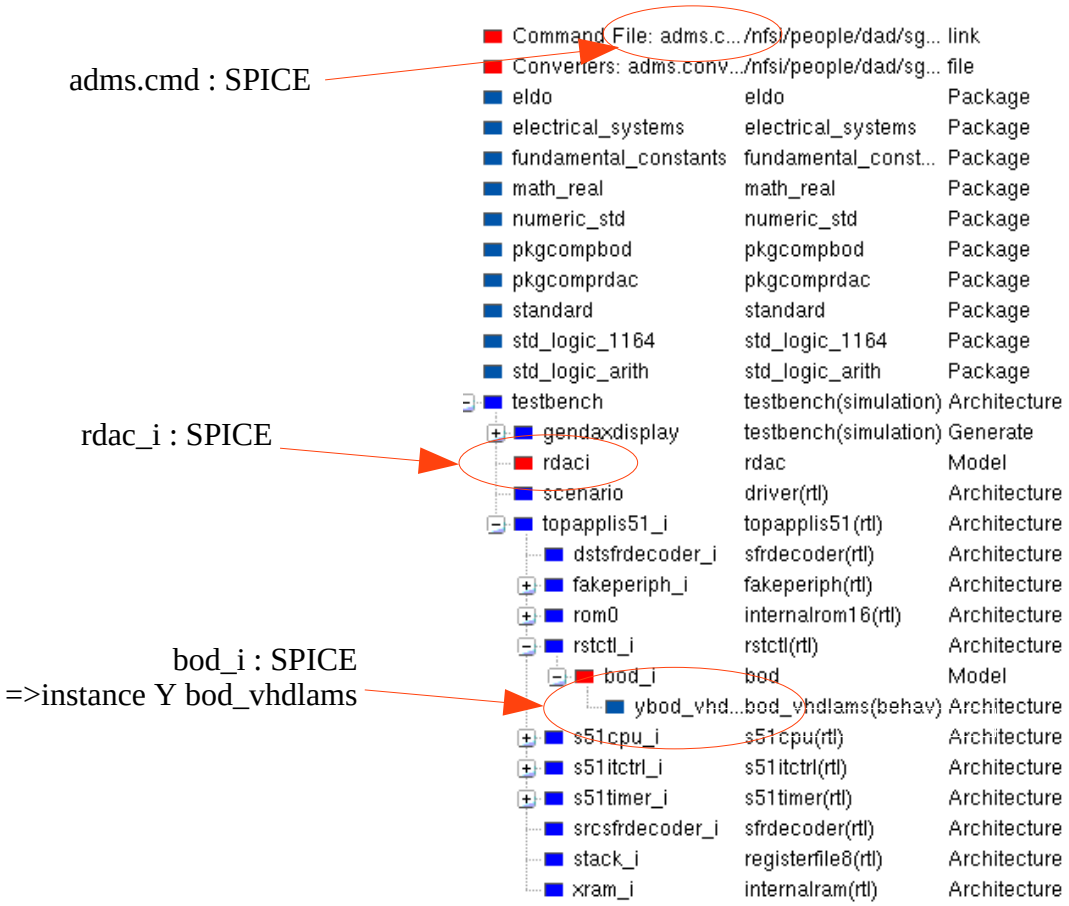
9.2 Structure de l'environnement de simulation

TOP VHDL, substitution de l'instance BOD_i du modèle BOD par un sous circuit SPICE qui instancie le modèle Bod_vhdlams.

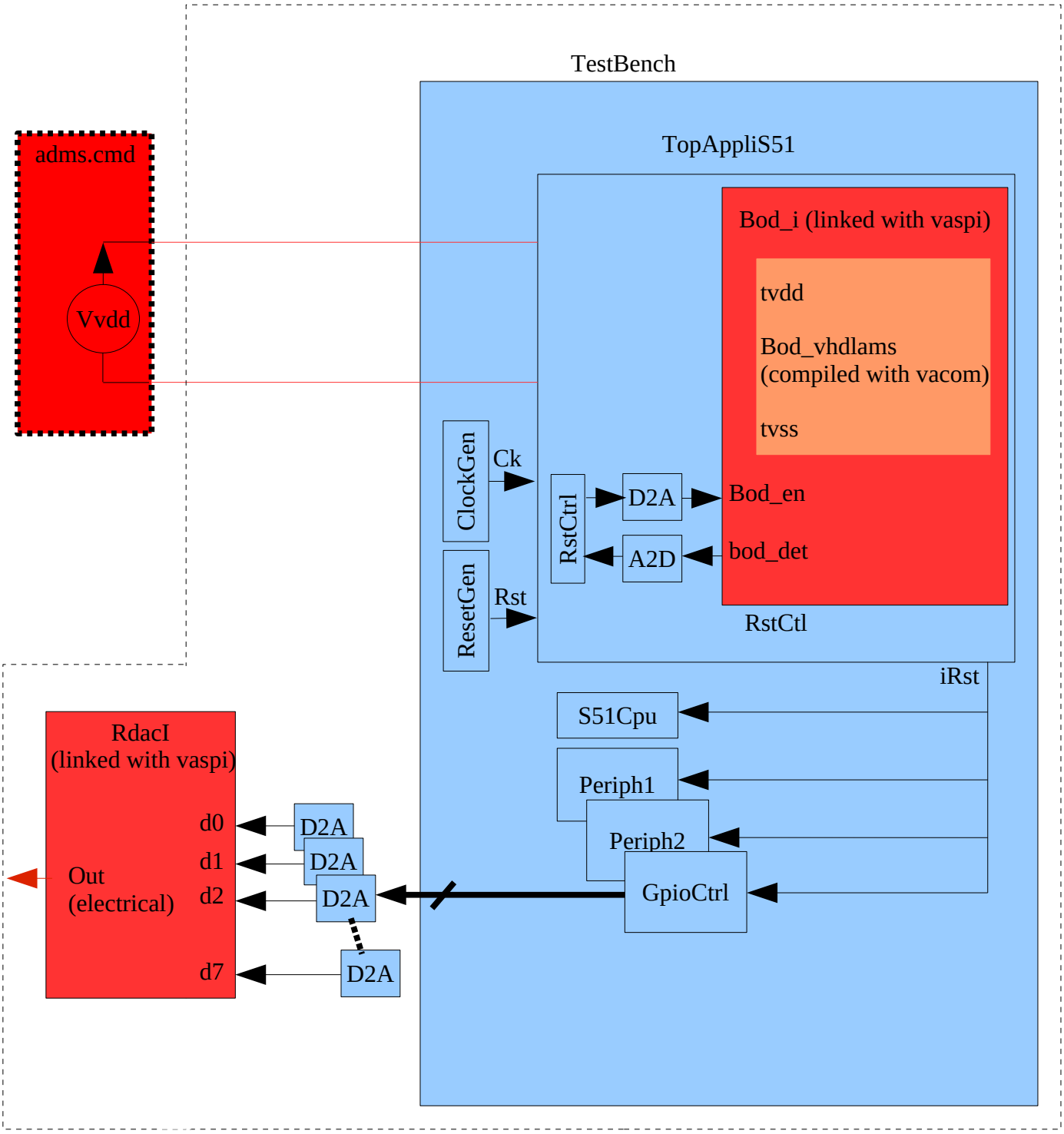
- Diagramme de flow:



- Vue structurelle:



- Vue bloc:


TestBench
adms.cmd
Vvdd
TopAppliS51
Bod_i (linked with vaspi)
tvdd
Bod_vhdlams
(compiled with vacom)
tvss
ClockGen
Ck
RstCtrl
D2A
A2D
Bod_en
bod_det
ResetGen
Rst
RstCtl
iRst
S51Cpu
Periph1
Periph2
GpioCtrl
RdacI
(linked with vaspi)
d0
d1
d2
d7
D2A
D2A
D2A
D2A
Out
(electrical)

9.3 Intégration du modèle de BOD en spice

9.3.1 Le testbench analogique

Il est défini dans le fichier *SCRIPTS/adms.cmd*.

- Définir la forme d'onde de l'alimentation du circuit à l'aide d'un PWL.

ex: *vdd vdd 0 5 PWL (0p 0 1400n 5 49992n 5 49994n 3 50u 5 55u 5 56u 3 80u 5)*

9.3.2 Importation du modèle de BOD en spice dans le projet

- Copiez votre netlist vers *TECH/SPICE/Bod/Bod.cir* (veillez à respecter les noms de terminaux du fichier existant : *bod_det* et *bod_en*)
- Si vous ne disposez pas de cette netlist, utilisez *TECH/SPICE/Bod/Bod_C35.cir*

9.4 Compilation (simulateur) avec ADVance MS

Attention il ne s'agit pas de la compilation du code applicatif.

Pour compiler la base complète avec Advance MS, allez dans le répertoire SCRIPTS et exécutez la commande suivante:

make compile_vams

Pour ne compiler que le modèle du BOD en VHDL-AMS, exécutez la commande suivante :

make compile_bod_vams

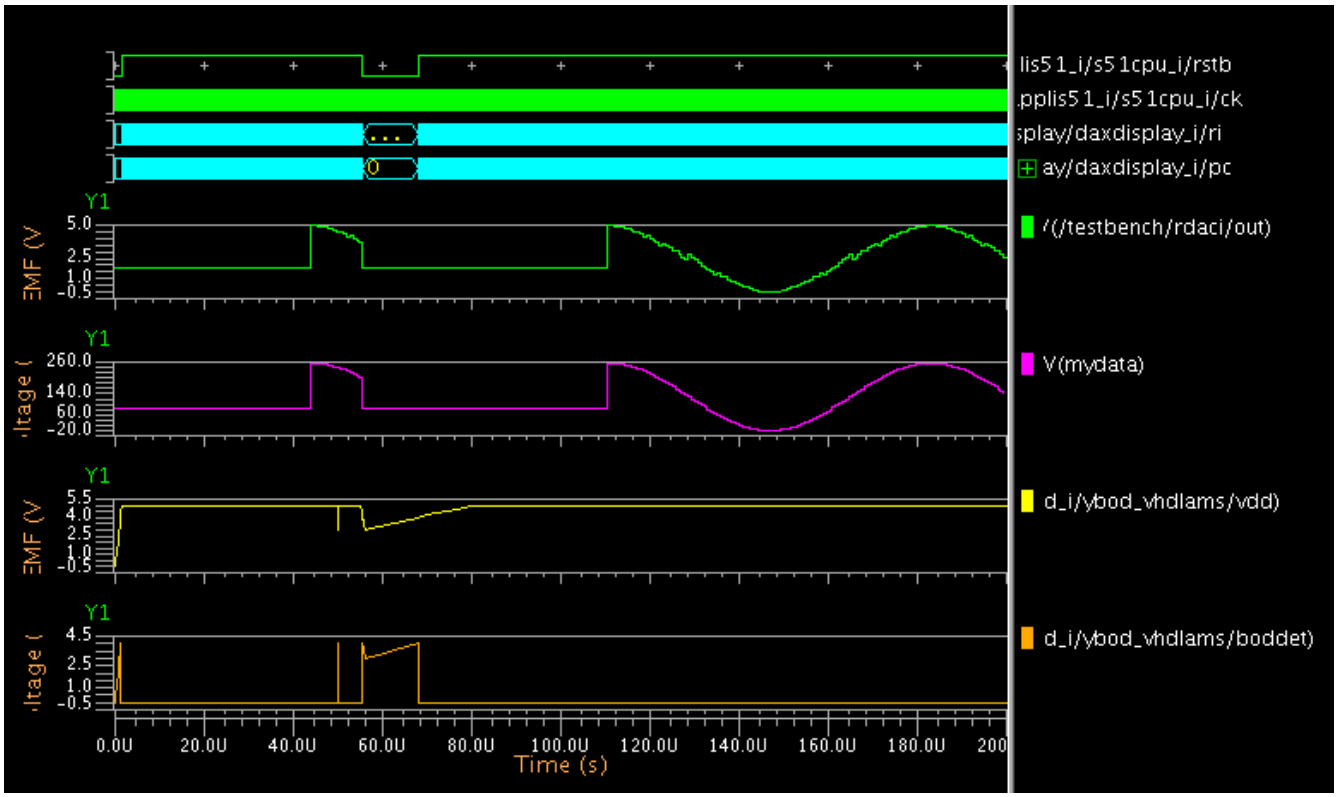
9.5 Simulation avec ADVance MS

Attention, la simulation se fera avec le dernier code applicatif compilé.

Dans le répertoire SCRIPTS, exécuter la commande:

make sim_adms

27 Simulation mixte, visualisation avec EZWAVE



10 Travaux pratiques

10.1 Consignes

10.1.1 Le premier jour

Constituez des groupes qui resteront **impérativement** les mêmes au cours des 5 séances.
Attention, il faut tenir compte de la journée « simulation mixte ».
Créez un compte gmail formé des noms des membres du groupe séparés par un « . » comme suit : NOM1.NOM2.NOM3@gmail.com
Communiquez par email le compte que vous avez créé à l'enseignant.
Celui-ci enverra une invitation par email à rejoindre le cours Google classroom « Projet de synthèse » en tant qu'élève qui servira à collecter les rapports de chaque séance.

10.1.2 A la fin de chaque séance

Vous devrez remettre un rapport par groupe qui sera noté.
Vous déposerez votre rapport dans l'espace classroom correspondant à la séance en cours.
Il est conseillé d'écrire votre rapport au fur et à mesure pendant les séances mais il pourra être rendu jusqu'à la veille de la séance suivante. Le document sera rendu **impérativement** au format PDF. Le nom du fichier devra respecter la convention suivante: **seance*Y_NOM1_NOM2*.pdf** avec :

- Y le numéro de séance
- NOM1, NOM2 les noms de famille du binôme
- Le caractère underscore (« _ ») utilisé comme séparateur

Il vous sera demandé d'apporter un soin particulier :

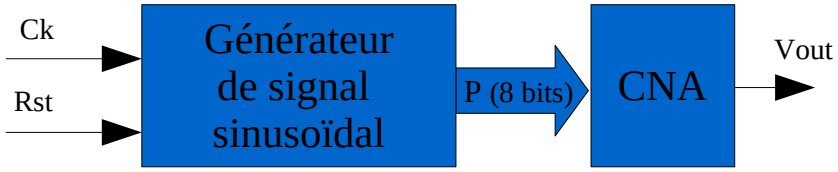
- au fond:
  - analyse du problème
  - analyse des solutions
  - réalisation
  - vérification
  - retour d'expérience
- à la forme
  - page de garde: titre, nom des étudiants du groupe, matière, département, date
  - le plan du document
  - copies d'écran (pas de photo)

10.2 Travaux Séance 1:

10.2.1 Cahier des charges

Objectif:
Concevoir un circuit ASSP générateur de signal sinusoïdal avec sortie numérique d'une résolution de 8 bits et de fréquence $\mathbf{f}_{signal}$ = 1Khz

28 Schémas bloc du circuit



10.2.2 Analyse de faisabilité

Réalisez un générateur de sinusoïde avec $\mathbf{f}_{signal\ =}$ 1Khz :

- Écrivez un code le plus simple possible en C pour imprimer les valeurs des échantillons dans la sortie standard de la console :
  - Dans le répertoire C du projet, prototypez sous UNIX un générateur C et générez la sinusoïdale de référence pour le reste de l'étude avec $\mathbf{f}_{éch}$=100KHz et $\mathbf{f}_{signal}$=1KHz.
  - Implémentez sur cible, pour cela utiliser le fichier existant gensin_simple.c dans le repertoire *SCRIPTS*. Référez vous à la section 3.2 « Flow de compilation en deux étapes (patch assembleur)» page 37.
- Le CPU fourni ne supporte qu'un sous ensemble du jeu d'instructions 51. Les instructions déjà implémentées sont les suivantes :

– *INC A* – *INC direct* – *LCALL addr16* – *RET* – *MOV A, direct* – *JC rel* – *ADD A,direct* – *MOV A, #data* – *ORL direct, #data* – *NOP* – *INC Rn*	– *MOV A, Rn* – *MOV direct, A* – *MOV Rn, #data* – *CJNE Rn, #data, rel* – *JNC rel* – *LJMP addr16* – *MOV direct, #data* – *MOVC A, @A+DPTR* – *SJMP rel* – *MUL AB*

Un code minimaliste nécessite au moins l'implémentation de l'instruction *MOV DPTR, #data16.* Vérifiez que votre code assembleur n'appelle pas d'autres instructions. Dans le cas contraire, simplifiez votre code.

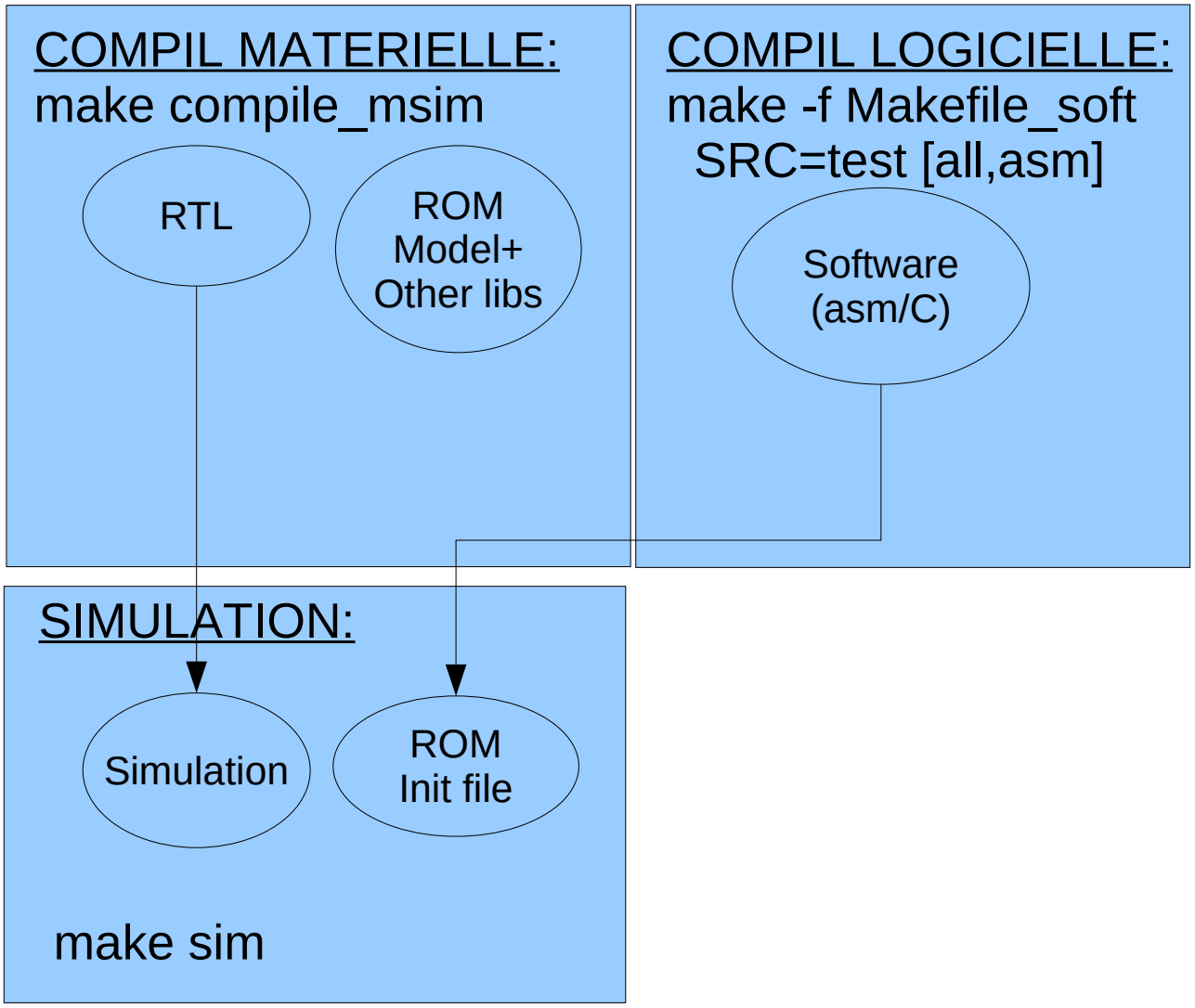
- Implémentez l'instruction *MOV DPTR, #data16 en vous référant à la section Cas du décodage de l'instruction : MOV DPTR, #DATA16* page *26.*
- Simulez en vous référant à la section 7 « Démarrer la simulation RTL »p°41.
- Implémentez l'instruction « *MOV direct1, direct2 » en vous référant aux sections 2.7.2 « Cas du décodage de l'instruction : MOV DIR, DIR» page 27 et 2.10.2 « Bus Périphérique (SFR)» page 33.*
- Remplacez l'instruction *« MOV direct, A »*dans votre code assembleur par l'instruction *« MOV direct1, direct2 »* et simulez.
- Obtenir la fréquence $\mathbf{f}_{signal}$ demandée en ajoutant des instructions NOP.

10.2.3 Éléments du rapport:

- Décrivez l'algorithme de génération de sinusoïde utilisé
- Décrivez les méthodes que vous avez utilisées pour obtenir f_{signal}=1KHz.
- Joignez le micro-code VHDL des instructions CPU implémentées.
- Retour d'expérience

Note : Respectez les consignes détaillées dans la section 10.1 page 50.

10.2.4 vue combinée des « flows »



10.3 Travaux Séance 2 :

10.3.1 Cahier des charges

Objectif:
Améliorer l'application précédente en utilisant une base de temps et en réduisant la consommation active du circuit.

10.3.2 Analyse

Une base de temps générant des événements système permet de rendre le signal de sortie indépendant de la fréquence d'exécution CPU, dans la mesure ou le CPU a le temps de réaliser ses tâches entre 2 événements consécutifs. Les moyens à mettre en œuvre sont un contrôleur d'interruption ainsi qu'un périphérique de type Timer générant des interruptions.

10.3.3 Implémentation du périphérique

- Étudiez la spécification section « Spécification du périphérique TIMER» p°66.
- Utilisez le squelette disponible dans le répertoire VHDL/S51Timer.
  Astuce : pour ne recompiler que le timer utilisez la commande suivante :
  make compile_tim
- Intégrez le périphérique dans le circuit en vous référant aux sections 11.4 « Intégration du périphérique TIMER »page 70 et 2.2.4 « Espaces d'adressage » page 19.
- Implémentez le périphérique Timer en *VHDL* (RTL). (note : ne pas implémenter la fonction TAI pour l'instant)
- Mettez à jour le CPU pour avoir le jeu d'instruction complet :
  cp ../VHDL/Misc/tpj2cpufile.txt ../VHDL/S51Cpu/DAX.vhdl
- Modifier l'application de la Séance 1 en utilisant la technique du polling et le timer (gensin_poll.c)
- Modifier l'application de la Séance 1 en utilisant le timer comme source d'interruption (gensin_it.c).
  - Configurez le contrôleur d'interruptions
  - Écrivez le handler C de l'IT du timer
  - Simulez
- Implémentez et intégrez la fonction TAI.
- Modifier l'application de la Séance 1 en utilisant le timer en mode « auto idle » (gensin_tai.c)

10.3.4 Éléments du rapport

- Décrivez les algorithmes de génération de sinusoïdale que vous mettez en place.
- Décrivez les avantages et les inconvénients des méthodes « interruption » et « auto idle ».

10.4 Travaux Séance 3 :

10.4.1 Cahier des charges

Objectif:
Obtenir une netlist de portes logiques standard AMS du circuit ainsi qu'un fichier de timing au format SDF et savoir les analyser.
Ouils: RTL Compiler (CADENCE)

10.4.2 Synthèse logique du périphérique timer

- Réalisez la synthèse du timer conçu lors de la séance précédente (voir le script SCRIPTS/synth_timer.tcl et la section 4 « Démarrer la synthèse logique» p°38).
- Analyse de la netlist:
  - Analysez les types de cellules logiques utilisées.
  - Décrivez les cônes logiques aboutissant aux entrées *d* et *en* de la bascule D *ROLLV* (cf. schémas section 11.3.3 «Interface utilisateur» p°67. (noms des instances, des nets et des composants)
  - Lancez plusieurs synthèses en batch avec des fréquences d'horloge différentes (5MHz, 20MHz, 100MHz, 150MHz, 250MHz, 500MHz, 1GHz …) et tracez sur des graphes la surface et le temps de propagation du chemin le plus long en fonction de la fréquence visée à partir des fichiers de sortie de l'outil de synthèse.
    Note : Le nombre de porte logiques équivalent est obtenu en divisant la surface du design avec la surface d'une porte logique Nand20 dans la technologie AMS C35 ($54.6um^2$).
  - Décrivez le chemin critique du Timer en réalisant une analyse statique de timing sur la synthèse à 1GHz.
- Bonus : Réaliser 2 synthèses avec 2 implémentations de machine d'état différentes :
  - Version avec état futur codé en dur
  - Version avec état futur par défaut donné par le registre d'état présent

10.4.3 Synthèse logique du circuit

- Réalisez la synthèse du circuit complet :
  make synth_chip
- Référez vous à la section 4 « Démarrer la synthèse logique» p°38.
- Annotez le schéma de la section 11.3.3 p°67 avec les noms de net, d'instance et de composants de la netlist.

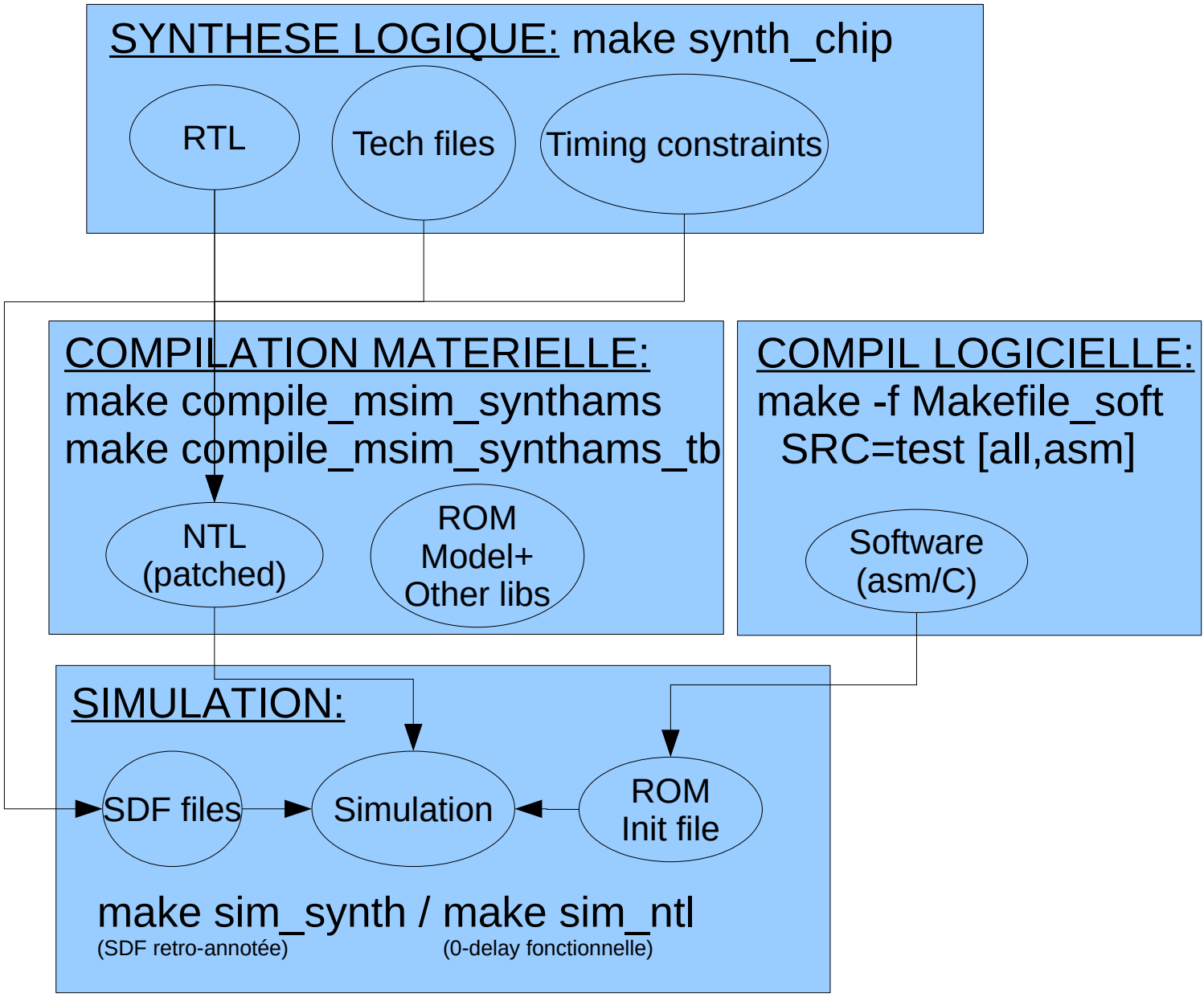
10.4.4 Simulation NTL sans timing du circuit

- Réalisez les simulations post-synthèse des applications gensin_poll.c et gensin_it.c
  Référez vous à la section 8 « Démarrer la simulation NTL» p°42 et à la section 10.4.6 « vue combinée des « flows » p°55.

10.4.5 Simulation NTL rétro-annotée du circuit

- Analysez le fichier de timings sdf dans le corner max et annotez le schéma de la section 11.3.3 p°67 avec les temps d'interconnexion et les temps de traversée intrinsèque des cellules.
- Corréler avec une simulation temporelle.
- Réalisez les simulations post-synthèse rétro-annotées en timing des applications gensin_poll.c et gensin_it.c. Référez vous à la section 8 « Démarrer la simulation NTL» p°42.
- Analyse des timings:
  - Mettre en évidence sur un chronogramme issu d'une simulation les temps de propagation des arcs C→Q et C→QN de la bascule ROLLV
  - Expliquez le comportement temporel du registre *Dout0I* du périphérique *GpioCtrl*.

10.4.6 vue combinée des « flows »



10.5 Travaux Séance 4 :

10.5.1 Cahier des charges

Objectif: Optimiser le circuit en fréquence.

10.5.2 Analyse du rapport surface/fréquence de fonctionnement

- Lancez plusieurs synthèses avec les fréquences d'horloge suivantes : 20MHz puis 1MHz puis 32MHz en suivant les étapes ci-dessous:
  - modifiez la période de la clock *sys_clk* dans *synth.tcl*
  - *make synth_chip* (les fichiers de synthèse seront sauvegardés sous *SCRIPTS/SYNTFILES)*
- Indiquez le temps de propagation du chemin le plus long et la surface totale en fonction de la **fréquence de synthèse**.
- A l'aide de simulations rétro-annotées, mettre en évidence la limitation en **fréquence (de simulation)** du circuit:
  - Simulez *gensin_poll.c* avec la netlist 1MHz avec clock externe à 32MHz
  - Simulez *gensin_poll.c* avec la netlist 20MHz avec clock externe à 32MHz
  - Simulez *gensin_poll.c* avec la netlist 32MHz avec clock externe à 32MHz

    note1 : aidez-vous du rappel des commandes de la section 10.4.6 « vue combinée des « flows » » p°55.

    note2 : pour simuler une netlist précédemment synthétisée, il faut recopier les fichiers de synthèse dans l'espace de compilation :

    cp SYNTFILES/clock_period_<clock_period>_<time>/top_synth_ams. ../NTL*

10.5.3 Analyse statique de timing

- A l'aide de *RTL compiler* en mode graphique faites une analyse statique du chemin le plus long du circuit sur la netlist issue de la synthèse à 32MHz
  - Si nécessaire, recopiez les fichiers de la synthèse 32MHz (voir note2 du paragraphe précédent) dans le repertoire NTL
  - *make synth_chip_restore* (relit le .db du repertoire NTL)
  - *gui_show* (commande pour lancer le GUI à partir du shell RTL compiler)
  - dans le menu : timing → report → bargraph (cf. section 6 « Analyse statique de timings » p°39)
  - simuler *div.asm* et mettre en évidence les délais

10.5.4 Optimisation en timing/surface

- L'instruction DIV nécessite un opérateur de division complexe. Dans l'hypothèse où cette instruction serait peu utilisée mais néanmoins nécessaire, quelle solution serait envisageable pour réduire le nombre de portes logiques pour une synthèse à fréquence max ? (référez-vous à la section 11.5 « Instruction DIV » p°71).
  - Évaluez cette solution (gain en surface, slack du cost group « ALU_DIV » , limitations fonctionnelles).
  - Faites une simulation netlist rétro-annotée du circuit modifié avec div.asm à 32MHz.
- Dans l'optique d'un circuit spécialisé et optimisé au maximum en coût, on peut retirer du circuit des fonctions logiques inutilisées (voir section 2.6 Architecture du cœur S51 (vue détaillée) p°25). Évaluez le gain en surface de cette solution par rapport à la solution « general purpose ».

Note : Un décodeur d'instruction optimisé est décrit dans le fichier *../VHDL/Misc/tpj4cpufile.txt*
Vous pouvez l'utiliser en lieu et place de *../VHDL/S51Cpu/DAX.vhdl*
Pensez à sauvegarder votre décodeur avant, vous en aurez besoin en 10.5.5. (*cp ../VHDL/S51Cpu/DAX.vhdl ../VHDL/S51Cpu/DAX.vhdl.sav*)

10.5.5 Optimisation du circuit en consommation

- Si vous avez précédemment écrasé votre *DAX.vhdl* (voir note de la section précédente) alors restaurez le (*cp ../VHDL/S51Cpu/DAX.vhdl.sav ../VHDL/S51Cpu/DAX.vhdl*)
- Faites une synthèse du circuit avec l'option :
  set_attribute lp_insert_clock_gating true
  (enlever le # à la ligne 43 du script *synth.tcl*).
  Mettre en évidence les dispositifs de réduction de consommation insérés par le synthétiseur en analysant les cônes logiques contrôlant les bascules D du module GpioCtrl. Expliquez le fonctionnement de ces dispositifs et vérifiez le à l'aide d'une simulation.

10.5.6 Éléments du rapport

- Décrivez les chemins longs que vous avez analysés sous forme de schémas de portes logiques inter-connectées.
- Présentez tous vos résultats d'essais de synthèse et commentez les.

10.6 Travaux Séance 5 :

10.6.1 Cahier des charges

Objectif: Vérifier les interfaces entre les domaines analogique et digital.

10.6.2 Simulation mixte

- Écrivez le modèle VHDL-AMS du superviseur de tension utilisé. Référez vous à la section 11.6 « Spécification du bloc BOD» p°72 et section 9.4 « Compilation (simulateur) avec ADVance MS».
- Familiarisez vous avec l'environnement de simulation mixte. Référez vous à la section 9 « Démarrer la simulation mixte» p°44.
- Simulez le circuit avec votre modèle comportemental de superviseur dans les scénarios suivants: (écriture d'un *PWL* dans le fichier *adms.cmd*, référez-vous à la section 9.3.1 « Le testbench analogique » p°48.)

  Note1 : *RstB* est appliqué 1.5us dans *Testbench.vhdl*, ce qui simule l'appui sur un bouton poussoir reset. Pour jouer sur l'ordre du relâchement du bouton reset et de la montée d'alim, jouez sur le PWL de l'alimentation dans *adms.cmd*.

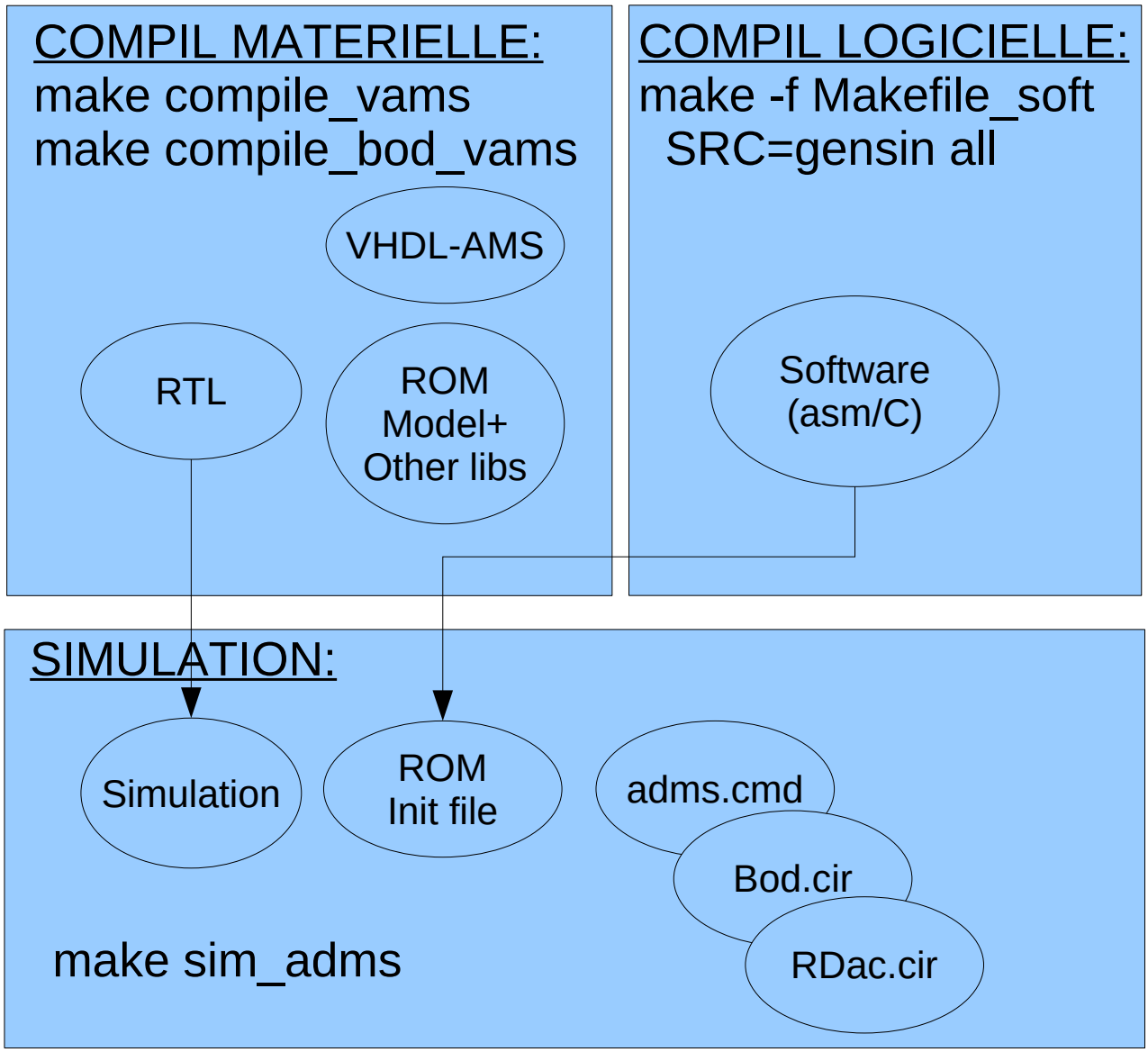
  Note2 : Pensez à initialisez votre environnement : . ams_x35

  - Démarrage 1 :
    - Appliquer un reset externe
    - Montée d'alimentation
    - Relâcher le reset externe après que l'alimentation a atteint le seuil de détection du BOD
  - Démarrage 2 :
    - Appliquer un reset externe
    - Montée d'alimentation
    - Relâcher le reset externe avant que l'alimentation atteigne le seuil VIL des cellules standard de la technologie C35 (2.5V).
  - Démarrage 3 :
    - Montée d'alimentation
    - Simuler une chute de tension à 2.7V pendant l'exécution de l'application
- Simulez le circuit avec la netlist spice du superviseur dans les mêmes scénarios. (voir section 9.3.2 « Importation du modèle de BOD en spice dans le projet».
  - Ce superviseur de tension peut-il être utilisé pour faire un reset du circuit en phase de démarrage ?
  - Comparez le comportement des modèles VHDL-AMS et SPICE.
- Écrivez le modèle *spice* d'un « R/2R ladder DAC » sans l'amplificateur de sortie. Référez vous à la section 11.8 « R/2R ladder DAC» p°75 et simulez.
  - Pourquoi la sortie du DAC est-elle bruitée ?

10.6.3 Éléments du rapport

- Représentez les chronogrammes des trois cas de figure énoncés précédemment.
- Comparez les traces de sortie du modèle de BOD en *vhdl-ams* et en *spice*.

10.6.4 vue combinée des « flows »



11 Annexe

11.1 Structure du répertoire projet

C	.....	Prototypage haut niveau de l'application
DOC	.....	Documentation
NTL	.....	Netlist VHDL et fichier SDF après synthèse
SCRIPTS	.....	Scripts (compilation C, ASM, simu RTL, synthèse)
	Analyse	Analyse du jeu d'instructions minimal
	Lib.	Fichiers de définition des SFRs pour compilation C et ASM
	Tools	Compilateurs, outils de conversion
TECH	.....	
	dolphin_ram32k8	RAM 32Kx8 sur la techno ciblée
	dolphin_rom32k8	ROM 32Kx8 sur la techno ciblée
	GTECH	Modèles verilog des cellules GTECH
	AMS	AMS standard cells technology files
	SPICE	Modèles spice de blocs analogiques (ex: BOD)
	VHDLAMS	Modèles VHDL-AMS de blocs analogiques (ex: BOD)
	VHDL	Modèle VHDL de blocs analogiques (ex: BOD)
	ELDO	Modèles ELDO de base de la technologie AMS
VHDL	.....	
	GpioCtrl	General Purpose I/O controller
	InternalRam	RAM utilisateur
	InternalRom16	ROM code
	Pkg	Package de fonctionsVHDL utiles
	PkgS51Cpu	Package de configuration du coeur du S51
	PkgS51ItCtrl	Package de configuration du contôleur d'ITs
	RegisterFile8	Banc de registres système (Stack S51, 128 bytes)
	S51Cpu	Cœur de micro 51 pipeliné
	S51ItCtrl	Contrôleur d'ITs
	S51Timer	Timer
	SfrDecoder	Décodeur d'adresse SFR
	TestBench	Testbench
	TopAppliS51	Top circuit
WORK	.....	Work simulation RTL
WORK_NTL	.....	Work simulation NTL (permet de tourner les simus en //)
WORK_ADMS	.....	Work simulation mixte avec ADMS

11.2 Jeu d'instruction complet

Signification des symboles utilisés:

Rn	Un des registres actifs R0 à R7
direct	Adresse d'un octet de RAM interne, d'un port, ou SFR
@Ri	Adressage indirect par registre R0 ou R1
#data	Donnée 8 bits
#data16	Donnée 16 bits
bit	Adresse au niveau du bit
rel	Adresse relative au PC en complément à 2 de -128 à +127
addr11	Adresse limitée au bloc de 2Ko dans lequel figure l'instruction
addr16	Adresse sur l'espace de 64Ko

- L'instruction MOV : (MOV <dest>, <source>)

Mnémonique	Syntaxe	Code	octets	cycle
MOV	A, Rn	E8+n	1	1
MOV	A, direct	E5	2	1
MOV	A, @Ri	E6+i	1	1
MOV	A, #data	74	2	1
MOV	Rn, A	F8+n	1	1
MOV	Rn, direct	A8+n	2	2
MOV	Rn, #data	78+n	2	1
MOV	direct, A	F5	2	1
MOV	direct, Rn	88+n	2	2
MOV	direct, direct	85	3	2
MOV	direct, @Ri	86+i	2	2
MOV	direct, #data	75	3	2
MOV	@Ri, A	F2+i	1	1
MOV	@Ri, direct	A6+i	2	2
MOV	@Ri, #data	76+i	2	1
MOV	DPTR, #data16	90	3	2

- L'instruction MOVC : (MOVC A, @A+<base-reg>) permet de lire un octet dans la mémoire pgm.

MOVC	A,@A+DPTR	93	1	2
MOVC	A,@A+PC	83	1	2

- L'instruction MOVX : (MOVX <dest>, <source>) permet la lecture ou l'écriture d'un octet en RAM externe.

MOVX	A,@Ri	E2+i	1	2

MOVX	A,@DPTR	E0	1	2
MOVX	@Ri,A	F2+i	1	2
MOVX	@DPTR,A	F0	1	2

- Les instructions PUSH et POP permettent respectivement de sauvegarder sur la pile ou d'y récupérer des données.

PUSH	direct	C0	2	2
POP	direct	D0	2	2

- L'instruction XCH : (XCH A, <byte>) échange les données de l'accumulateur A et de l'octet adressé.

XCH	A, Rn	C8+n	1	1
XCH	A, direct	C5	2	1
XCH	A, @Ri	C6+i	1	1

- L'instruction XCHD : (XCHD A, @Ri) échange les quartets de poids faible entre l'accu A et l'octet adressé.

XCHD	A, @Ri	D6+i	1	1

- L'instruction ADD : (ADD A, <byte>) additionne un octet et l'accu A, résultat dans A.

ADD	A,Rn	28+n	1	1
ADD	A, direct	25	2	1
ADD	A, @Ri	26+i	1	1
ADD	A, #data	24	2	1

- L'instruction ADDC : (ADDC A, <byte>) additionne un octet, l'accu A et la retenue, résultat dans A.

ADDC	A, Rn	38+n	1	1
ADDC	A, direct	35	2	1
ADDC	A, @Ri	36+i	1	1
ADDC	A, #data	34	2	1

- L'instruction SUBB : (SUBB A, <byte>) soustrait un octet ainsi que la retenue au contenu de A, résultat dans A.

SUBB	A, Rn	98+n	1	1
SUBB	A, direct	95	2	1
SUBB	A, @Ri	96+i	1	1
SUBB	A, #data	94	2	1

- L'instruction INC : (INC <byte>) incrémente un octet ou DPTR.

INC	@Ri	06+i	1	1
INC	A	04	1	1
INC	direct	05	2	1
INC	Rn	08+n	1	1
INC	DPTR	A3	1	2

- L'instruction DEC : (DEC <byte>) décrémente un octet.

DEC	@Ri	16+i	1	1

DEC	A	14	1	1
DEC	direct	15	2	1
DEC	Rn	18+n	1	1

- L'instruction MUL : (MUL AB) multiplie l'accu A et le registre B, résultat : octet faible dans A et octet fort dans B.
- L'instruction DIV : (DIV AB) divise le contenu de A par le contenu de B, quotient dans A et reste dans B.
- L'instruction DA : (DA A) ajustement décimal de A.

MUL	AB	A4	1	4
DIV	AB	84	1	4
DA	A	D4	1	1

- L'instruction ANL : (ANL <dest>, <source>) réalise un ET logique entre source et dest, résultat dans dest.

ANL	A, #data	54	2	1
ANL	A, @Ri	56+i	1	1
ANL	A, direct	55	2	1
ANL	A, Rn	58+n	1	1
ANL	direct, #data	53	3	2
ANL	direct, A	52	2	1
ANL	C, /bit	B0	2	2
ANL	C, bit	82	2	2

- L'instruction ORL : (ORL <dest>, <source>) réalise un OU logique entre source et dest, résultat dans dest.

ORL	A, #data	44	2	1
ORL	A,@Ri	46+i	1	1
ORL	A, direct	45	2	1
ORL	A, Rn	48+n	1	1
ORL	direct, #data	43	3	2
ORL	direct, A	42	2	1
ORL	C, /bit	A0	2	2
ORL	C, bit	72	2	2

- L'instruction XRL : (XRL <dest>, <source>) réalise un OU exlusif logique entre source et dest, résultat dans dest.

XRL	A, #data	64	2	1
XRL	A, @Ri	66+i	1	1
XRL	A, direct	65	2	1
XRL	A, Rn	68+n	1	1
XRL	direct, #data	63	3	2
XRL	direct, A	62	2	1

- L'instruction CLR : (CLR <A/bit>) met A ou un bit à 0

CLR	A	E4	1	1

CLR	bit	C2	2	1
CLR	C	C3	1	1

- L'instruction CPL : (CPL <A/bit>) complémente A ou un bit

CPL	A	F4	1	1
CPL	bit	B2	2	1
CPL	C	B3	1	1

- L'instruction RL : (RL A) rotation vers la gauche du contenu de A
- L'instruction RLC : (RLC A) rotation vers la gauche du contenu de A+retenue
- L'instruction RR : (RR A) rotaion vers la droite du contenu de A
- L'instruction RRC : (RRC A) rotation vers la droite du contenu de A+retenue
- L'instruction SWAP : (SWAP A) échange le quartet de poids faible avec celui de poids fort de A

RL	A	23	1	1
RLC	A	33	1	1
RR	A	03	1	1
RRC	A	13	1	1
SWAP	A	C4	1	1

- L'instruction SETB : (SETB <bit>) met à 1 un bit

SETB	bit	D2	2	1
SETB	C	D3	1	1

- L'instrucion MOV : (MOV <bitdest>, <bitsrc>) copie le bitsrc dans le bitdest

MOV	bit, C	92	2	2
MOV	C, bit	A2	2	1

- L'instruction ACALL : réalise un saut absolu inconditionnel
- L'instruction LJMP : réalise un saut long inconditionnel
- L'instruction SJMP : réalise un saut court par adressage relatif
- L'instruction JMP : réalise un saut indirect

AJMP	addr11	E1	2	2
LJMP	addr16	02	3	2
SJMP	rel	80	2	2
JMP	@A+DPTR	73	1	2

- L'instruction JZ : saut si A=0
- L'instruction JNZ : saut si A<>0
- L'instruction JC : saut si retenue à 1
- L'instruction JNC : saut si retenue à 0
- L'instruction JB : saut si bit à 1
- L'instruction JNB : saut si bit à 0
- L'instruction JBC : saut si le bit est à 1 et mise à zero de celui-ci

JZ	rel	60	2	2
JNZ	rel	70	2	2
JC	rel	40	2	2
JNC	rel	50	2	2

JB	bit, rel	20	3	2
JNB	bit, rel	30	3	2
JBC	bit,rel	10	3	2

- L'instruction CJNE : (CJNE <byte1>, <byte2>, <rel>) saut si byte1 et byte2 sont différents

CJNE	@Ri, #data, rel	B6+i	3	2
CJNE	A, #data, rel	B4	3	2
CJNE	A, direct, rel	B5	3	2
CJNE	Rn, #data, rel	E8+n	3	2

- L'instruction DJNZ : (DJNZ <byte>, rel) décrémente byte et saut si résultat différent de 0

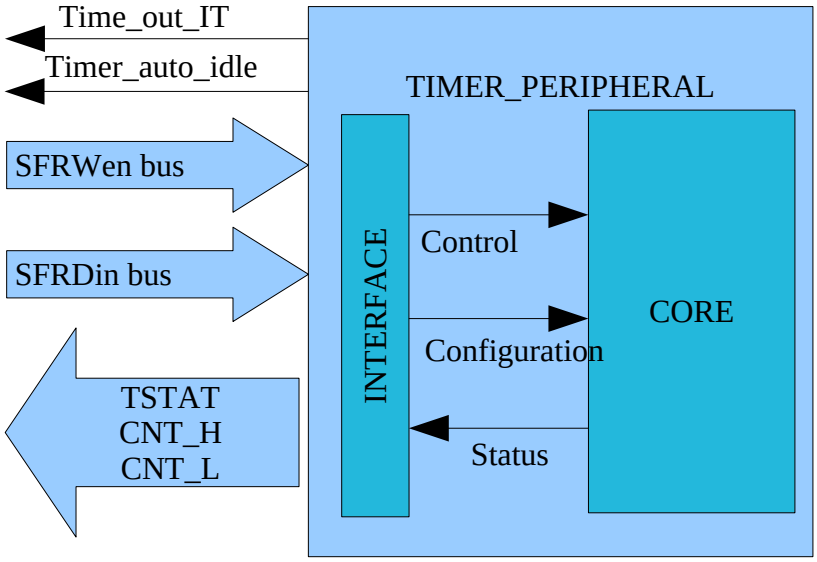
DJNZ	direct, rel	D5	3	2
DJNZ	Rn, rel	D8+n	2	2

- L'instruction NOP : pas d'opération

NOP	-	00	1	1

11.3 Spécification du périphérique TIMER

11.3.1 Schéma bloc



11.3.2 Entrées/Sorties

Signal	Type	Bus size	Desc
SFRWrite	I	1	Indique une écriture sur le bus
TCTRLWen	I	1	Sélection de TCTRL pour écriture
CNT_HWen	I	1	Sélection de CNT_H pour écriture
CNT_LWen	I	1	Sélection de CNT_L pour écriture
SFRDin	I	8	Bus data en entrée (écriture)
Time_out_IT	O	1	Signal d'IT sur niveau
Timer_auto_idle	O	1	Signal de requête de mode IDLE
TSTAT	O	8	Sortie du registre TSTAT
CNT_H	O	8	Sortie du registre CNT_H
CNT_L	O	8	Sortie du registre CNT_L

11.3.3 Interface utilisateur

- registres de contrôle:

Registre TCTRL: registre de contrôle

Bit number	7	6	5	4	3	2	1	0
Field name	-	-	-	-	AUTOIDLE	CLRSTAT	STOP	START
Access	-	-	-	-	W	W	W	W
Rst value	-	-	-	-	-	-	-	-

START: Démarre la state-machine du timer
STOP: Stoppe la state machine du timer, retour à l'état initial
CLRSTAT: Ecrire un 1 remet à 0 le bit de status ROLLV (la mise à 1 matérielle est prioritaire sur la mise à 0 logicielle)
AUTOIDLE: Ecrire un 1 endort le CPU jusqu'au prochain débordement de compteur (la requête de réveil du débordement de compteur est prioritaire) sur la mise à 1 logicielle.
La lecture de ce registre retourne une valeur non définie.

- registres de configuration

Registre CNT_H: borne max (poids forts) du compteur interne

Bit number	7	6	5	4	3	2	1	0
Field name	CNT_H							
Access	RW							
Rst value	0							

Registre CNT_L: borne max (poids faibles) du compteur interne

Bit number	7	6	5	4	3	2	1	0
Field name	CNT_L							
Access	RW							
Rst value	0							

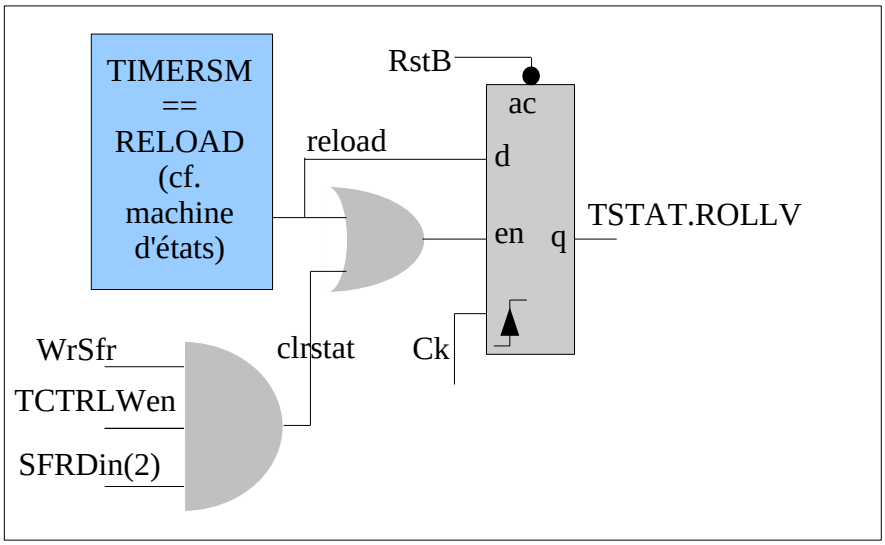
- registres de status

Registre TSTAT: Status du périphérique

Bit number	7	6	5	4	3	2	1	0
Field name	-							ROLLV
Access	-							R
Rst value	-							0

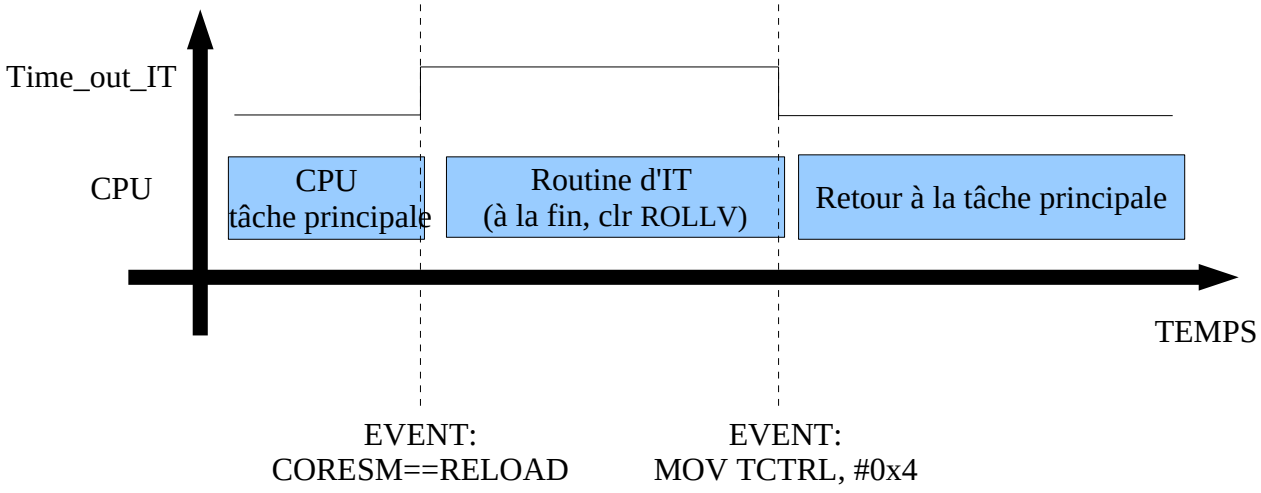
Indique qu'un débordement de compteur s'est produit, ce status peut être relu par interface SFR (permet le polling software) via le CPU, de plus cette DFF peut être câblée sur une entrée du contrôleur d'IT. Ce bit est mis à 0 en écrivant un '1' dans le bit CLRSTAT du registre TCTRL. La mise à 1 du registre par le matériel (state machine) est prioritaire sur le CLEAR (logiciel) pour ne pas rater d'événements.

Exemple d'implémentation du registre de status:

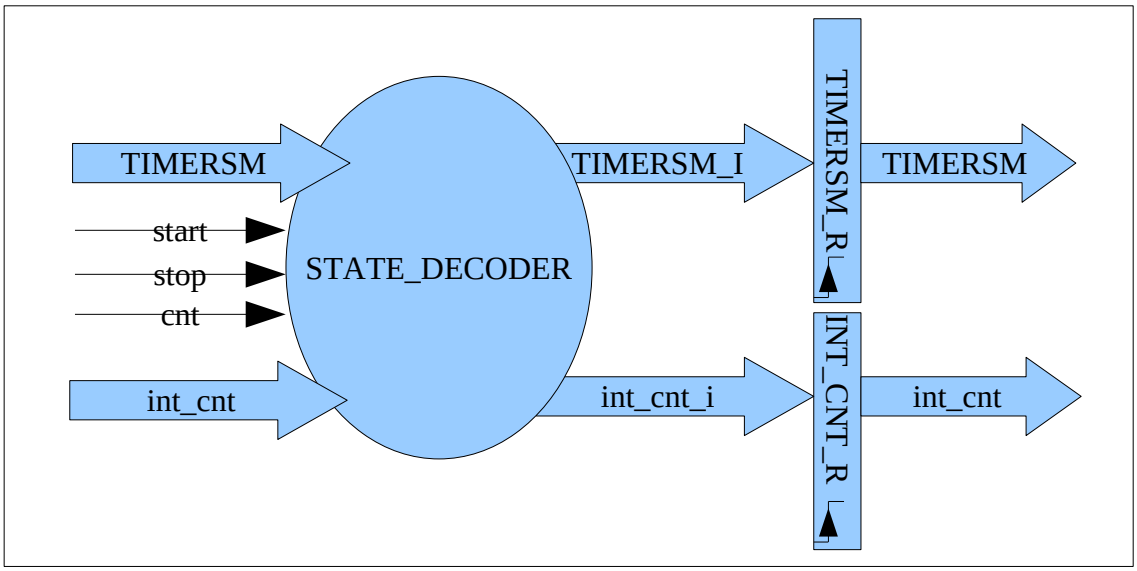


11.3.4 Étude des transactions CPU/Périphérique lors d'une interruption

Time_out_IT reste actif tant que le bit ROLLV du registre TSTAT n'a pas été remis à 0 par le CPU.

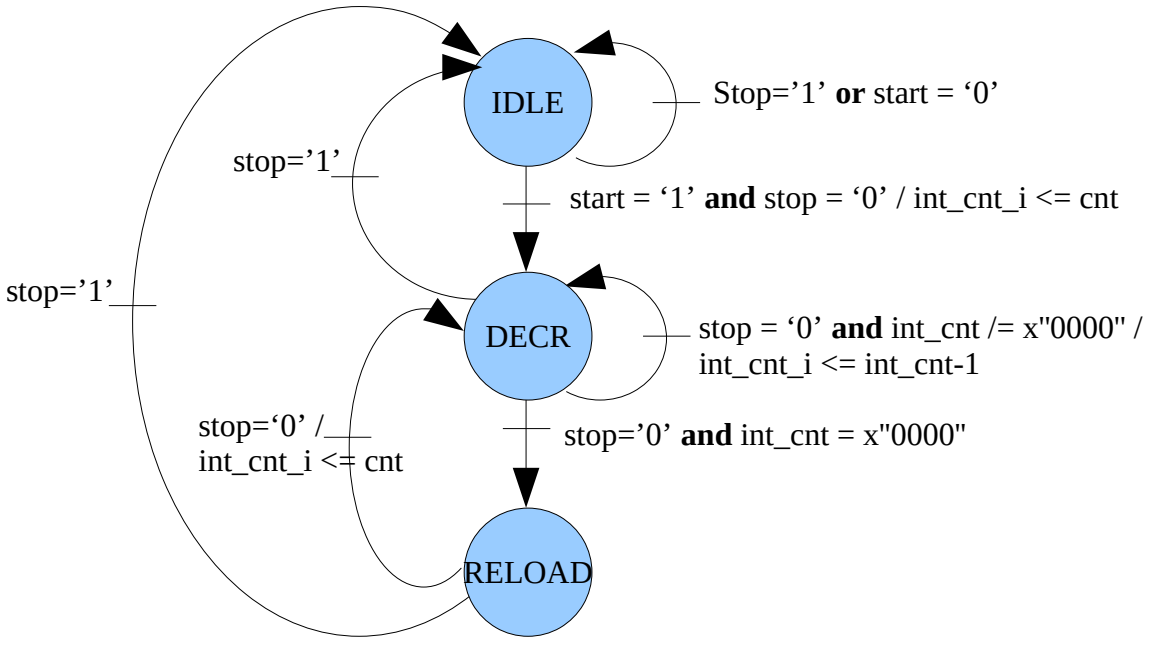


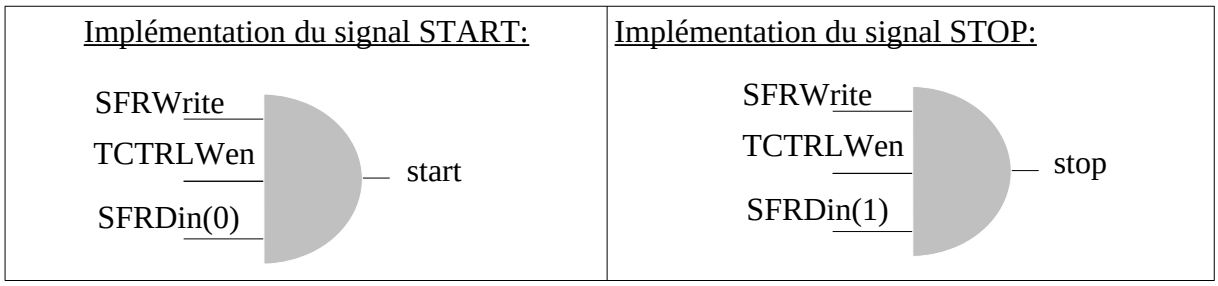
11.3.5 Schéma bloc



11.3.6 Machine d'état du cœur du périphérique

ressource interne: Registres d'état + logique combinatoire de la machine d'états finis+registre de comptage interne de 16 bits.





11.4 Intégration du périphérique TIMER

Ce périphérique s'intègre sur :

1. le bus de communication système (SFR)

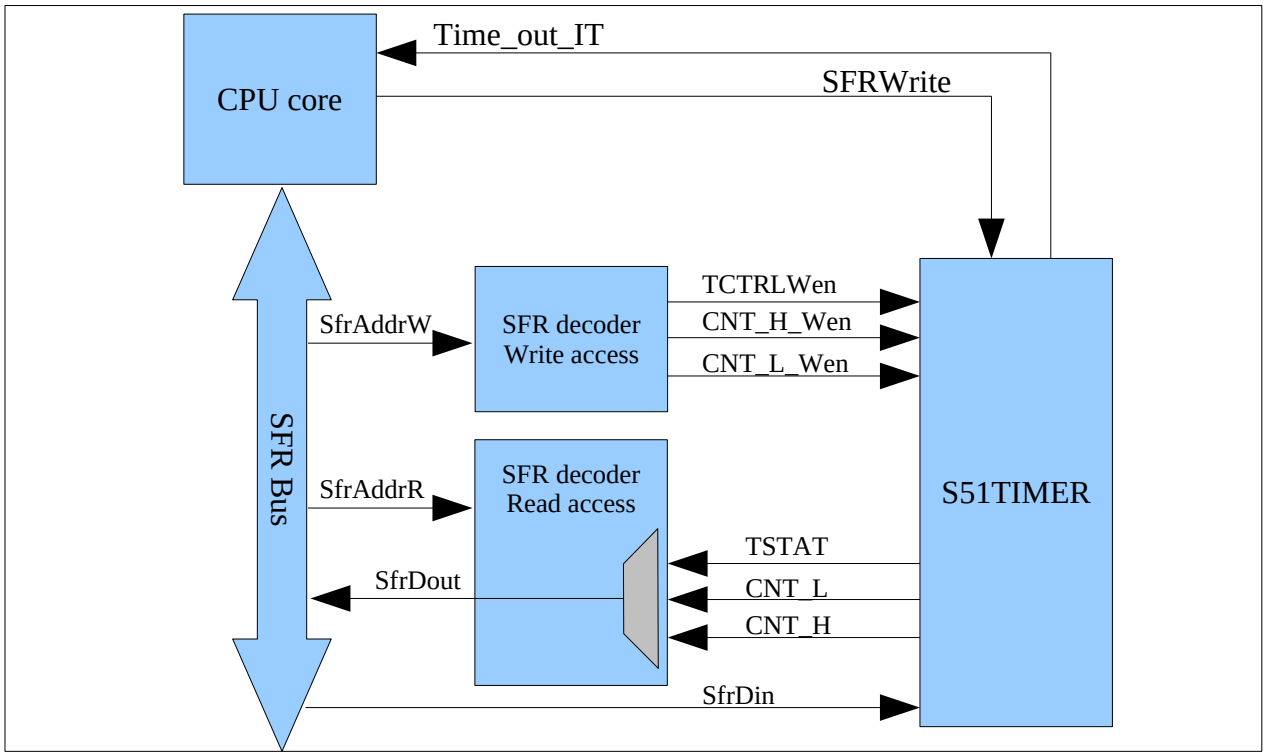
Chaque registre du périphérique a une adresse qui est définie au niveau du circuit. Le décodage d'adresses est réalisé dans le bloc *SfrDecoder*.

Pour plus de détails, reportez vous au sections suivantes:

- *section 2.2.4 « Espaces d'adressage » page 19.*
- *section 2.3 « Architecture système » page 21.*
- *section 2.10.2 « Bus Périphérique (SFR)» page 33.*

L'intégration sur le bus SFR se fait :

- en rajoutant le démultiplexage des signaux de sélection des registres du périphérique (voir section 11.3.3 « Interface utilisateur » page 67.
- en câblant les signaux de contrôle et de données du bus SFR du périphérique à ceux du CPU *(SFRRead, SFRWrite, SFRDout, SFRDin).*



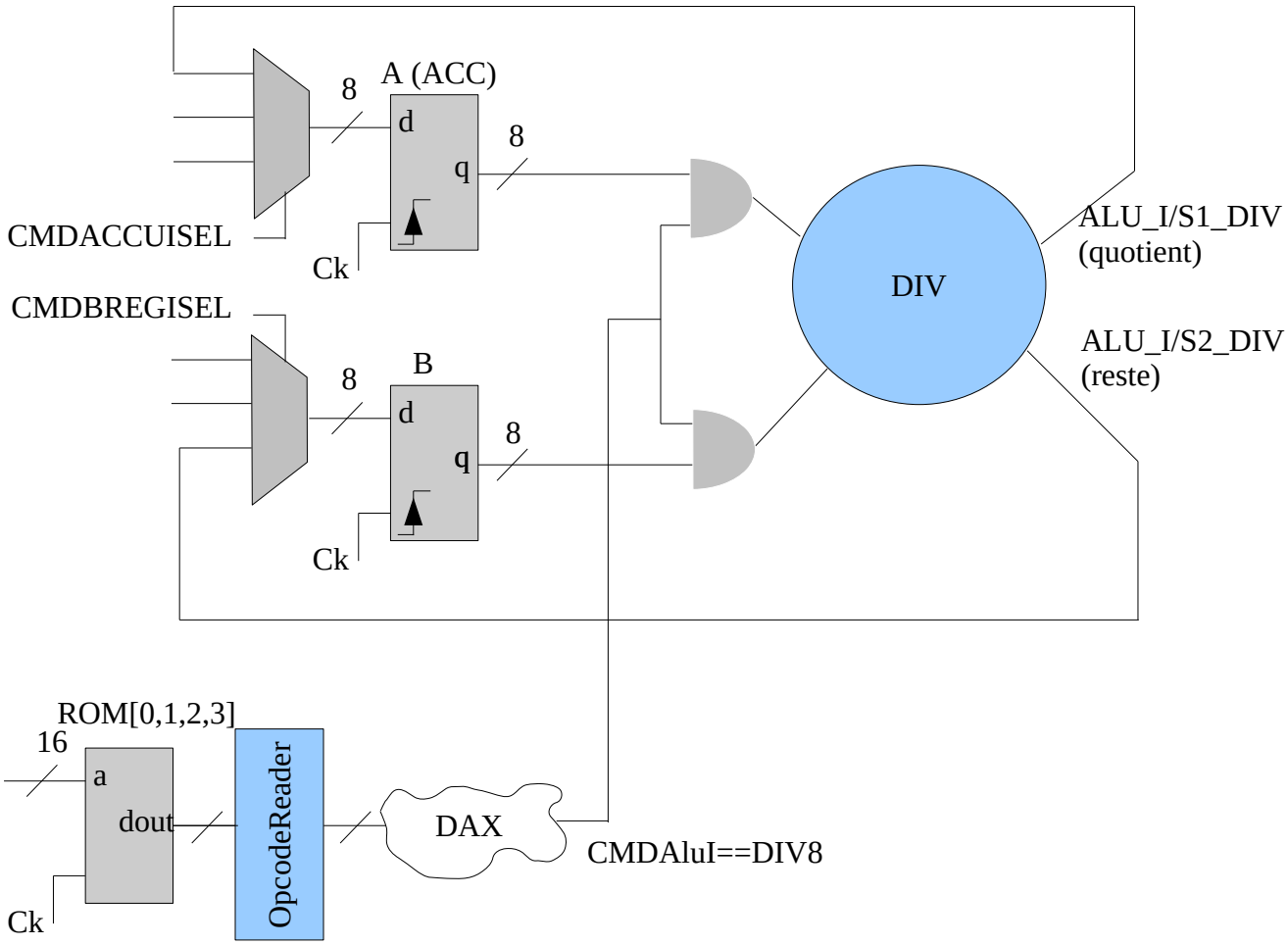
2. le contrôleur d'interruptions

Reportez vous à la section 2.8 « Les interruptions » page 28.

Connectez la source d'interruption du S51TIMER (*Time_out_IT*) à la ligne 0 du contrôleur d'interruption ItEntriesV(0).

11.5 Instruction DIV

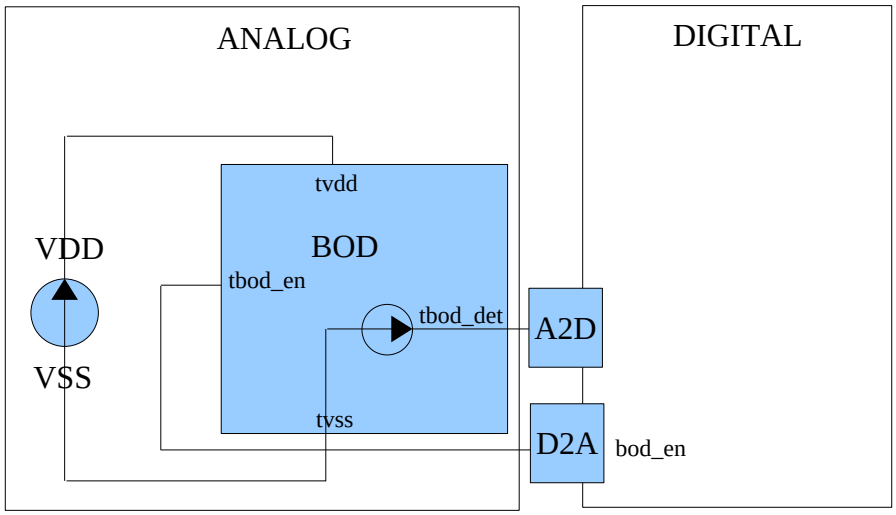
L'instruction DIV : (DIV AB) divise le contenu de A par le contenu de B, quotient dans A et reste dans B.



11.6 Spécification du bloc BOD

11.6.1 Description générale

Ce bloc est un BOD très simple sans calibration ni hystérésis ni contrôle spécifique pour sélectionner une source de tension de comparaison. Une entrée *enable* commande la source de courant du bloc. Il n'y a pas de mécanisme d'hystérésis intégré, on compensera l'absence de stabilisation de la sortie par un dispositif digital adéquat.



11.6.2 Description des terminaux

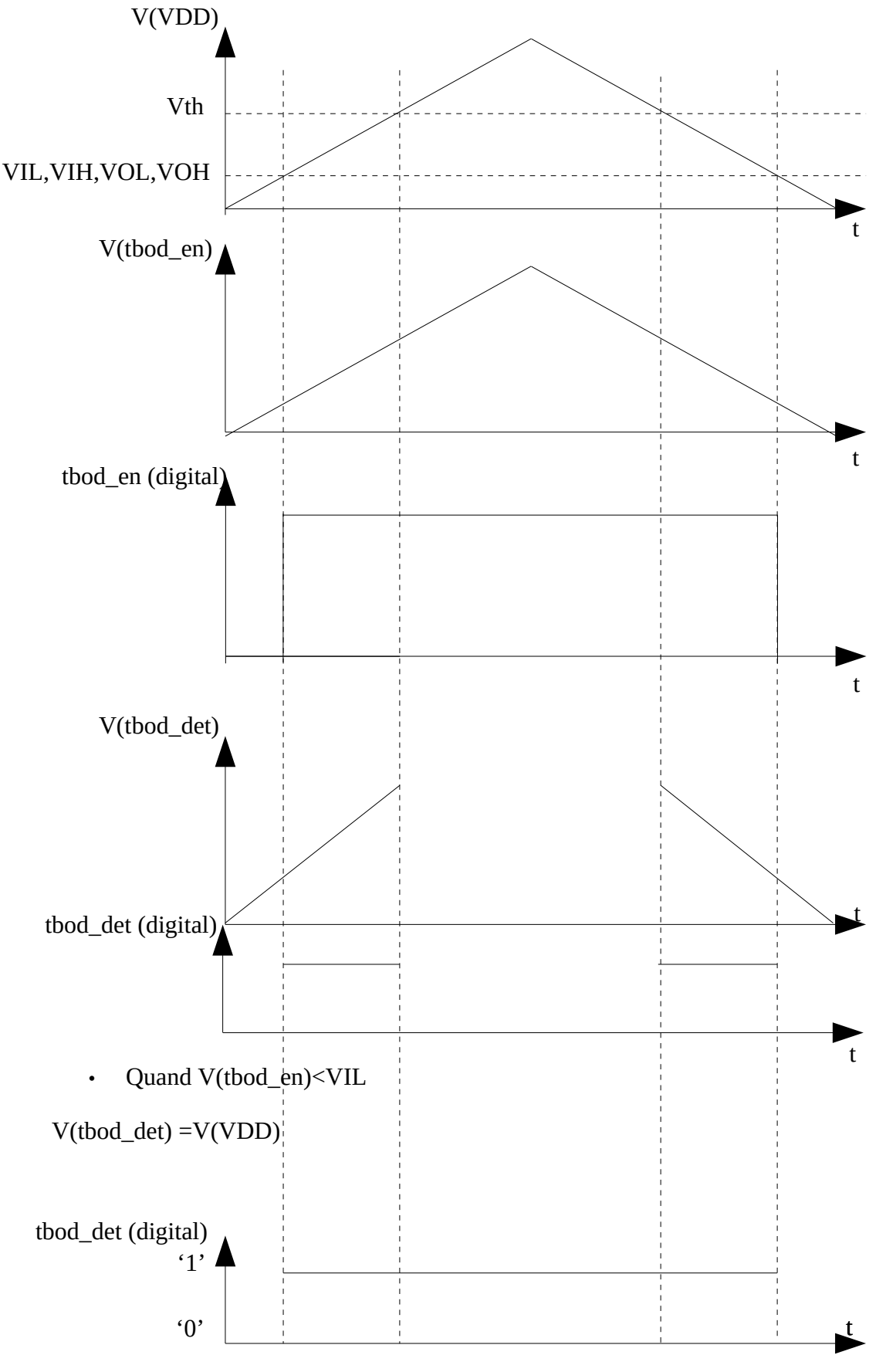
Nom de la pin	Type	Fonction
tvdd	Analog (input)	Alimentation (du circuit) à surveiller
tvss	Analog (input)	Masse du circuit
tbod_det	Analog (output)	Sortie du superviseur de tension: V(tvdd): quand V(tbod_en)<VIL ou V(tvdd)<Vth (Vth=4.0V) V(tvss) : quand V(tbod_en)>VIH et V(tvdd)>=Vth (Vth=4.0V)
tbod_en	Analog (input)	V(tbod_en)>VIH: BOD activé V(tbod_en)<VIL: BOD désactivé

Notes :

- Les niveaux VIL, VIH, VOL, VOH des cellules standard de la technologie C35 sont égaux à 2.5V.
- L’impédance d’entrée du bloc entre les terminaux tvdd et tvss est considérée infinie
- L’impédance d’entrée du bloc entre les terminaux tbod_en et tvss est considérée infinie

11.6.3 Fonction

- Quand V(tbod_en)>=VIH



- Quand V(tbod_en)<VIL

V(tbod_det) =V(VDD)

11.7 Spécification du bloc RSTCTL

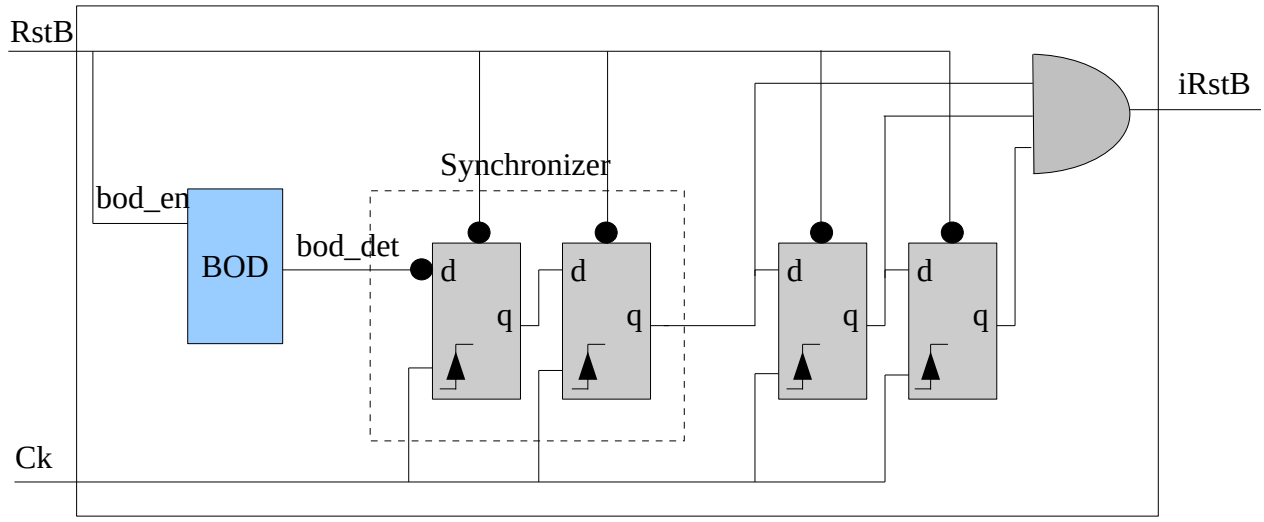
11.7.1 Description

Ce bloc génère un *reset* interne au circuit en lorsque l'entrée *reset* externe est active ou que le BOD detecte une chute de tension. Au démarrage le *reset* externe doit être relâché après l'établissement de l'alimentation. En effet le bloc RSTCTL n'intègre pas de POR qui aurait permis le démarrage automatique du circuit.

Le bloc RSTCTL embarque une structure de synchronisation pour synchroniser la sortie du BOD dans le domaine d'horloge Ck (pour éviter le phénomène de méta stabilité (cf. section 1.6.4 « Méta-stabilité » p°11)).

Une structure de registres à décalages permet ensuite avec une porte « ou » d'allonger le signal de reset interne qui est très chargé et de le relâcher sur front montant pour respecter les temps recovery/removal de l'entrée reset asynchrone des bascules par rapport à l'entrée clock (reset appliqué de manière asynchrone et relâché de manière synchrone).

11.7.2 Architecture

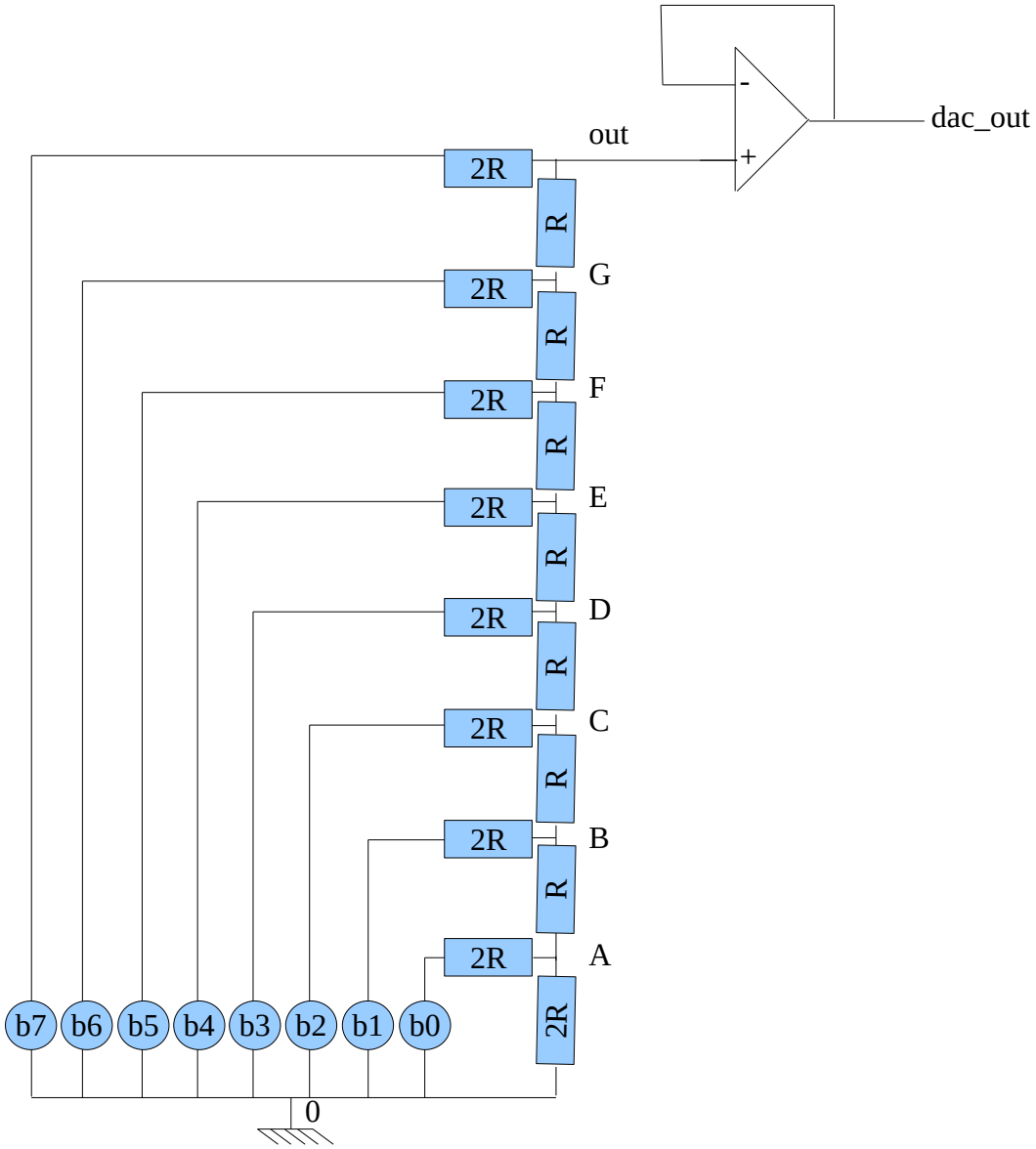


11.7.3 Entrées/Sorties

Signal	Type	Bus size	Desc
Ck	I	1	Horloge système
RstB	I	1	Reset externe
iRstB	O	1	Reset interne (BOD+RstB), relâchement synchrone

11.8 R/2R ladder DAC

11.8.1 Schémas de principe



11.8.2 Schémas équivalent Thévenin