Rapport de stage

Sébastien LAUNAY
DESS CISAN
Année 2003

Elaboration de moyens de tests pour l'acquisition électronique numérique de l'expérience de physique EDELWEISS II

Rapport de stage

Sébastien LAUNAY
DESS CISAN
Année 2003

Elaboration de moyens de tests pour l'acquisition électronique numérique de l'expérience de physique EDELWEISS II

Commissariat à l'Energie Atomique
DSM DAPNIA SEDI
91191 Gif-sur-Yvette cedex

Université Pierre et Marie Curie
DESS CISAN
LIP6/ASIM
12 rue Cuvier – 75005 PARIS

Responsable de stage :
Monsieur Hervé Deschamps

Responsable de DESS :
Madame Anne Derieux

Sommaire

Remerciements

Je tiens, en premier lieu, à remercier Monsieur Philippe Rebourgeard, chef du service SEDI (Service de l'Electronique, des Détecteurs et de l'Informatique) et Monsieur Michel Mur, chef de laboratoire TRAPS (TRaitement des données d'Acquisition et de Processeurs Spécialisés) pour m'avoir accueilli au sein de leur équipe.

Merci à Monsieur Hervé Deschamps, tuteur de stage, ingénieur chercheur, pour m'avoir permis d'intégrer le projet EDELWEISS, ainsi que pour avoir effectué mon suivi au sein de l'entreprise.

Je remercie également l'ensemble des ingénieurs et techniciens du département DAPNIA (Département d'Astrophysique, de physique des particules, de Physique Nucléaire et de l'Instrumentation Associée) et plus particulièrement du service SEDI, pour leur disponibilité et leurs conseils, et qui ont contribué à rendre cette période de stage des plus intéressante et agréable.

Introduction

Depuis les années 1930, les physiciens cherchent à résoudre une énigme surprenante connue sous le nom de masse manquante de l'Univers. La masse visible des étoiles et gaz interstellaires semble en effet ne former qu'une faible partie (environ 10 %) de la masse estimée des galaxies.

Quelle est donc cette matière invisible qui n'émet ni n'absorbe de lumière ? Plusieurs candidats ont été supposés (Neutralinos, Machos,…). Les théories supersymétriques de la physique des particules prévoient l'existence de particules massives, appelées WIMPs (Weakly Interacting Massive Particules) excellents candidats pour cette masse manquante.

Plusieurs expériences sont en cours pour la détection de la matière noire ; parmi celles-ci, EDELWEISS (Expérience pour DEtecter Les Wimps En SIte Souterrain) est dédiée en particulier à la recherche du WIMP (Weakly Interacting Massive Particules) et fournit pour l'année 2002 le meilleur résultat mondial.

Plusieurs laboratoires du CEA, participent à l'expérience EDELWEISS au sein d'une collaboration qui regroupe différents organismes (CNRS, IN2P3, INSU, SPM, CSNSM et CRTBT). C'est donc dans le domaine de la recherche fondamentale, et dans le cadre de cette collaboration que s'effectue le déroulement du stage, au sein du service SEDI (Service de l'Electronique, des Détecteurs et de l'Informatique) du département DAPNIA (Département d'Astrophysique, de physique des particules, de Physique Nucléaire et de l'Instrumentation Associée) de la DSM (Direction des Sciences de la Matière).

Dans le but de connaître d'avantage cet environnement de travail, ce document présente en premier lieu, les diverses activités du CEA ; pour ensuite décrire plus précisément le contexte du stage autour de l'expérience EDELWEISS II, successeur à plus grande échelle de l'expérience actuelle. Ce qui amènera à expliciter les différentes activités réalisées durant cette période ; activités dont le rôle a été de satisfaire les besoins grandissants en précision et en sensibilité de l'expérience.

1 Qu'est-ce que le CEA ?

Le CEA est un acteur clef de la recherche, du développement et de l'innovation en matière d'énergie, de défense, de technologies de l'information, ainsi que de santé.

Depuis sa création en 1945, il relève des défis scientifiques majeurs dans un ensemble de domaines : programmes électronucléaires, dissuasion nucléaire, micro et nanotechnologies, astrophysique, imagerie médicale, toxicologie ou biotechnologies…

Pour l'ensemble de ses recherches, le CEA dispose en outre, d'outils exceptionnels et particulièrement performants (supercalculateur, réacteurs de recherches, grands instruments de la physique, lasers de puissance…).

Par ailleurs, toujours dans une volonté de mener une recherche scientifique de haut niveau, il entretient d'étroites relations avec les autres organismes de recherche et les universités, permettant une coordination des recherches, une mise en réseau des compétences et une optimisation des moyens.

Aujourd'hui, le CEA est reconnu comme un expert dans ses domaines de compétences, mais aussi, comme un moteur de l'innovation et de la diffusion technologique grâce à son implication dans le tissu industriel et économique et à de nombreux partenariats nationaux ou internationaux.

1.1 Assurer une énergie plus compétitive et respectueuse de l'environnement

Une politique énergétique inscrite dans le développement durable doit à la fois permettre de répondre aux besoins croissants en énergie et de prévenir le réchauffement climatique : elle doit en particulier faire appel à toutes les énergies capables de répondre à ces enjeux, du nucléaire, apte à produire en masse et non émetteur de gaz à effet de serre, aux nouvelles énergies renouvelables.

Référence mondiale dans le domaine de la recherche électronucléaire, le CEA s'est donc engagé également dans le développement de nouveaux vecteurs énergétiques et de convertisseurs tels que l'hydrogène, les piles à combustible ou le photovoltaïque, par exemple.

Energie nucléaire : optimiser les systèmes actuels et élaborer le futur
Dans le domaine de l'électronucléaire, le CEA contribue à l'amélioration des performances du parc actuel de réacteurs et participe aux développements futurs.

Le CEA conduit ainsi des actions de recherche en partenariat avec les industriels pour répondre aux objectifs stratégiques d'allongement de la durée de vie des installations et d'amélioration des performances du combustible, incluant le retraitement/recyclage des combustibles usés afin d'en valoriser le contenu énergétique.

Il mène également des recherches sur les réacteurs et combustibles du futur avec comme objectif majeur une production électronucléaire très compétitive dont la sûreté sera encore améliorée, et la production de déchets minimisée.

En outre, le CEA étudie et développe des solutions techniques, efficaces et sûres pour la gestion des déchets radioactifs par la réduction de leur quantité et de leur nocivité et l'entreposage ou le stockage.

Le CEA poursuit parallèlement des recherches visant à mieux connaître les effets sur le vivant des rayonnements ionisants et des toxiques utilisés dans le nucléaire, afin d'apporter de nouvelles données pour l'évaluation scientifique des risques éventuellement associés aux activités nucléaires.

Améliorer la compétitivité des nouvelles sources d'énergie :
Le CEA consacre un important effort de recherche visant à rendre compétitives de nouvelles solutions énergétiques :

- Le programme hydrogène porte sur l'ensemble du cycle : production, stockage, utilisation, et distribution, pour trois grandes familles d'applications : les piles stationnaires, le transport, et les équipements portables en privilégiant deux filières de piles à combustibles : les piles à basse température à échange de protons (PEMFC), et celles à haute température à oxyde solide (SOFC).

- Dans le cadre du programme photovoltaïque, le CEA intensifie ses recherches dans les domaines des photopiles, des batteries, et des cellules solaires au silicium en film mince ou à base de matériaux organiques.

- Dans le domaine des économies d'énergie et de l'amélioration du rendement des systèmes de production d'énergie, le CEA met sa longue expérience de la modélisation numérique, des systèmes programmés intelligents, et de la micro-électronique appliqués aux équipements thermiques à haute efficacité, au service des nouvelles technologies du froid et à la réduction des pollutions dans les systèmes énergétiques.

Un enjeu de taille pour préparer le futur : la fusion nucléaire
La maîtrise de la fusion thermonucléaire contrôlée pourrait permettre, dans l'avenir, de disposer d'une source quasi infinie d'énergie. Le CEA est un acteur clef des recherches dans ce domaine, notamment par son implication dans les études du projet international ITER, étape essentielle vers la conception d'un réacteur de fusion électrogène.

1.2 Contribuer à la défense

Dans le cadre de la politique de dissuasion nucléaire française, le CEA est chargé de la conception, de la fabrication, du maintien en condition opérationnelle, et du démantèlement des têtes nucléaires. Par ailleurs, il conçoit et entretient les réacteurs nucléaires assurant la propulsion des bâtiments de la Marine nationale.

Pour la décennie à venir, le CEA a pour principale mission de mener à bien le programme *Simulation*, dont le but est de garantir à la France la pérennité de sa dissuasion après l'arrêt définitif des essais nucléaires.

Ce programme permettra de reproduire par le calcul les différentes étapes du fonctionnement d'une arme et de s'assurer ainsi de la fiabilité et de la sûreté des têtes nucléaires, sans avoir à les tester.

Dans ce cadre, le CEA s'équipe de supercalculateurs, outils indispensables à la simulation numérique et dispose aujourd'hui de la machine la plus puissante d'Europe.

Il met en oeuvre de nouveaux moyens de validation expérimentale tels qu'un laser de puissance, destiné à reproduire à échelle très réduite des conditions physiques proches de celles rencontrées lors du fonctionnement d'une arme nucléaire.

Enfin, le CEA contribue pour les instances nationales et internationales, à la surveillance du respect du traité d'interdiction complète des essais nucléaires (TICE) et à la lutte contre la prolifération.

1.3 Mettre la recherche technologique au service de l'industrie

En développant ses programmes de recherche nucléaire, le CEA a acquis des connaissances scientifiques lui permettant de développer parallèlement les technologies les plus innovantes pour l'industrie. Aujourd'hui, il est engagé principalement dans trois secteurs : les technologies de l'information, les matériaux et la santé.

Technologies de l'information
Les technologies de l'information font l'objet, au CEA, d'un important programme de R&D centré principalement sur les micro et nanotechnologies, l'instrumentation des systèmes complexes, et les technologies logicielles.

En micro et nanotechnologies, le CEA poursuit notamment des travaux sur la technologie Silicium sur Isolant (SOI), l'évolution des circuits intégrés (CMOS), ainsi que sur de nouveaux dispositifs innovants, comme les transistors à effet de champ .

En instrumentation et technologies logicielles, l'objectif est de constituer de nouveaux systèmes complexes interconnectés conciliant à la fois robustesse, fiabilité, confidentialité et convivialité. Le CEA travaille notamment sur la métrologie et le contrôle non destructif, la robotique interactive et les interfaces homme/machine (application des technologies de télé-opération à la chirurgie, par exemple) ainsi que sur les systèmes embarqués, notamment pour l'aéronautique.

Matériaux

Le CEA a également développé une expertise reconnue dans le domaine des matériaux et des procédés innovants selon deux approches :

- La solution des besoins à court terme vers l'industrie, avec pour objectif d'améliorer les processus de fabrication et de mise en forme des matériaux, le choix de leurs structures ainsi que leurs propriétés. Les études portent notamment sur la mise au point de nouveaux procédés d'assemblages résistants, tel que le soudage laser ou la réalisation de structures plus légères destinées au secteur des transports.

- L'identification de recherches amont à fort potentiel telles que les nanomatériaux qui reposent sur le développement d'approches multi-échelles et de simulation.

Technologies pour la santé

Les activités du CEA en matière de technologies pour la santé se concentrent essentiellement autour de l'imagerie médicale et des biotechnologies.

Pour l'imagerie médicale, il s'agit d'améliorer les performances des outils actuels tels que la gamma-caméra à semi-conducteur ou la radiographie numérique.

Dans le domaine des biotechnologies, le CEA développe différents outils, dont les biopuces - puces à ADN mais aussi laboratoires intégrés sur puces - ainsi que l'instrumentation associée. L'objectif est de mettre au point des microsystèmes capables de détecter, de diagnostiquer, ou encore d'analyser certaines fonctions génomiques.

Par ailleurs, depuis 2000, un test de dépistage de l'encéphalopathie spongiforme bovine développé par le CEA est utilisé par la plupart des pays européens dans le cadre de la campagne décidée par l'Union européenne.

Priorité au transfert de technologie et au partenariat
Pour l'ensemble de ces activités, la priorité est de privilégier les transferts vers le monde industriel. Ainsi le CEA est engagé dans de nombreux partenariats avec des entreprises françaises, européennes et internationales (ST Microelectronics, Motorola, Texas Instruments,…) pour le développement de nouvelles technologies.

1.4 Élargir les connaissances scientifiques

Indispensable à l'avancée des connaissances scientifiques, la recherche fondamentale joue un rôle central au sein du CEA. Outre les études amont dans les domaines de l'énergie, de la défense et de l'industrie, il poursuit différents programmes en sciences de la matière, sciences du vivant et sciences de l'environnement.

Sciences du vivant
Dans le domaine des sciences du vivant, la recherche concerne l'application des technologies issues du nucléaire aux domaines de la santé.
L'imagerie fonctionnelle, basé sur le marquage isotopique et la physique des rayonnements, permet d'explorer les organes de manière atraumatique. Le fonctionnement du cerveau humain, dans des tâches aussi spécifiques que le langage ou le calcul, peut ainsi être cartographié pour construire des modèles explicatifs. La compréhension détaillée du vivant passe aussi par l'analyse du produit des gènes au niveau de leur structure moléculaire ; la biologie structurale répond à ce besoin en combinant le marquage isotopique et l'utilisation des rayonnements, maîtrisés par le CEA.

Sciences du climat et de l'environnement
Issues de la recherche nucléaire, les méthodes de datation radioactives et les méthodes isotopiques ont permis aux équipes du CEA d'étudier depuis de nombreuses années les mécanismes climatiques. Les recherches portent à la fois sur la connaissance des climats passés mais aussi sur leur évolution future. Il s'agit notamment de chercher à mieux connaître les différents aspects du cycle du carbone et des composés à effet de serre pour les intégrer ensuite dans des modèles pronostiques globaux, ou encore, d'observer des climats passés grâce à l'analyse de prélèvements dans les glaces polaires ou dans les sols du fond des océans.

Sciences de la matière
Les recherches en sciences de la matière au CEA portent sur la structure de la matière, à petite et à grande échelle. Elles s'articulent autour de la physique des particules, la physique nucléaire, les nano sciences et l'astrophysique. Les recherches concernent en particulier les grands problèmes de la physique d'aujourd'hui comme la détermination de la nature de la masse cachée de l'univers ou la description des trous noirs. Le CEA collabore également à plusieurs programmes d'observation de l'Agence spatiale européenne (ESA).

C'est au sein de cette dernière section que s'effectue le stage qui exploite les connaissances scientifiques et plus particulièrement les sciences de la matière.

L'exploration de la matière offre, aujourd'hui encore, de larges champs d'investigation. Dans le domaine de la physique des particules, le CEA collabore avec le CNRS sur les trois grandes problématiques du domaine : la validation du modèle standard, la mise en évidence de la violation de symétrie matière-antimatière et le problème de la masse manquante de l'univers. Ainsi en 2002, la traque des « WIMPs » (Weakly Interacting Massive Particules) s'est resserrée avec l'expérience EDELWEISS (CEA/CNRS). Les chercheurs ont en effet exploré un domaine encore vierge du monde supersymétrique, ou l'on peut espérer capturer ces particules élémentaires très massives, et peut-être constitutives d'une partie de la matière noire.

2 L'expérience EDELWEISS

L'équipe du CEA (DRECAM, DAPNIA) est engagée dans la collaboration EDELWEISS (Expérience pour DEtecter Les Wimps En Site Souterrain) qui regroupe des équipes du CNRS (Centre National de la Recherche Scientifique) (IN2P3, INSU, SPM), du CSNSM (Centre de Spectrométrie Nucléaire et de Spectrométrie de Masse) et du CRTBT (Centre de Recherche pour les Très Basses Températures).

2.1 Que cherche-t-on ?

L'objectif est la détection directe de WIMPs, composants hypothétiques de la matière noire, au moyen de détecteurs ionisation-chaleur. Ces deux paramètres étant représentatifs des collisions de WIMPs sur le réseau cristallin des détecteurs ; ils permettent d'autre part, d'éliminer les événements crées par les rayons gamma ionisants. L'équipe participe à la mise en place de l'expérience au Laboratoire Souterrain de Modane (LSM) et aux prises de données.

2.2 Comment, quels sont les moyens mis en œuvre ?

Compte tenu de la difficulté de détection des WIMPs, il est indispensable d'isoler au maximum les détecteurs des rayonnements naturels. Ainsi, l'expérience EDELWEISS est protégée par 1600 mètres de roche sous le tunnel du Fréjus et les matériaux constitutifs de l'expérience sont rigoureusement sélectionnés pour leur basse radioactivité, ce qui conduit à une réduction par un facteur de 2 millions du flux de rayons cosmiques, et par un facteur 10000 du fond de neutrons.

La figure suivante permet d'évaluer la quantité de muons (particule élémentaire indésirable pour notre expérience) par rapport à la profondeur (en mètres ou en équivalent « eau ») ; ce qui démontre bien l'intérêt d'un laboratoire souterrain.

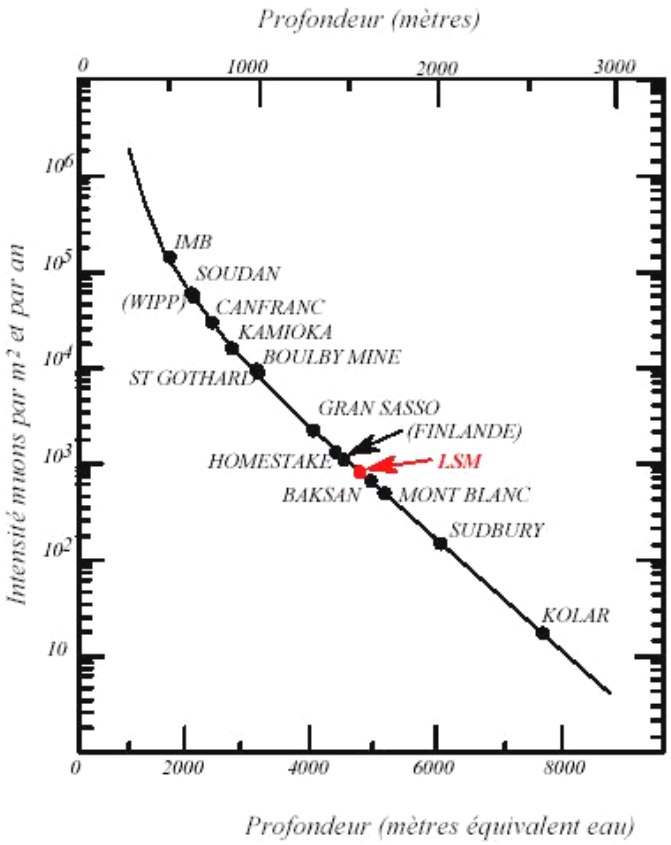
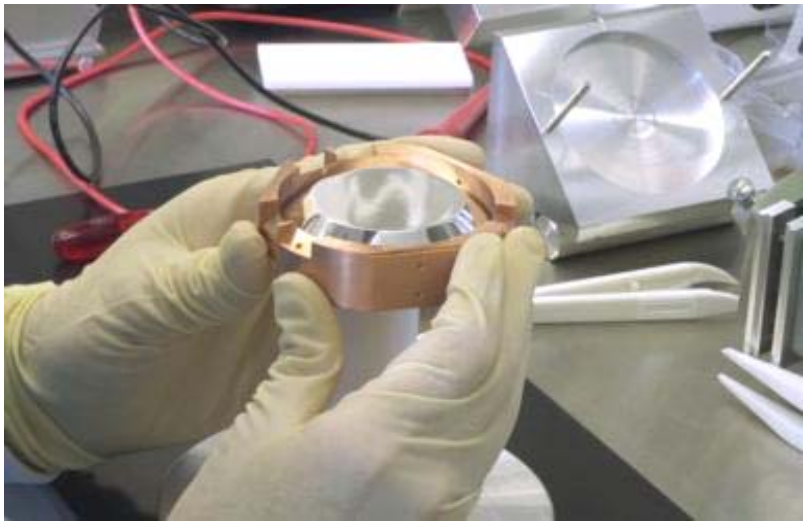


Figure 1 : Quantité de muons en fonction de la profondeur souterraine

Malgré ces précautions, un fond radioactif résiduel de rayons β, γ et de neutrons, persiste et il faut différencier l'impact d'un WIMP de celui du rayonnement résiduel. La mesure de l'énergie (par l'élévation de température) donne une évaluation de l'ensemble des interactions : les électrons et les photons des radioactivités β et γ sont essentiellement ionisants alors que les WIMPs induisent principalement un recul nucléaire (élévation de température).

L'expérience EDELWEISS est munie d'un système de double détection extrêmement sensible, par l'ionisation et par la chaleur : la première mesure permet d'enregistrer un signal de quelques centaines d'électrons, et la seconde, de mesurer une élévation de température d'un millionième de degré. Celle-ci est donnée par un thermomètre de type Ge NTD (Neutron transmutted doped Ge) collé sur le détecteur.

EDELWEISS utilise trois détecteurs de germanium ultra-pur de 320 grammes chacun (voir 2 et 3*)*, fonctionnant à une température de 20 millièmes de kelvin, proche du zéro absolu. Leur sensibilité leur permet de rejeter 99,9 % du bruit de fond radioactif.



Vue d'un détecteur de 320 grammes de germanium, construit au Dapnia/Sedi, dont les performances de rejet du bruit de fond radioactif de plus 99,9% ont permis à l'expérience EDELWEISS de tester une première région de modèles de supersymétrie, avec la meilleure sensibilité mondiale.

Figure 2 : Détecteur de 320 grammes

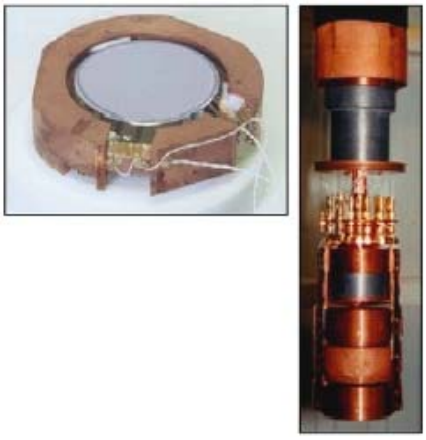
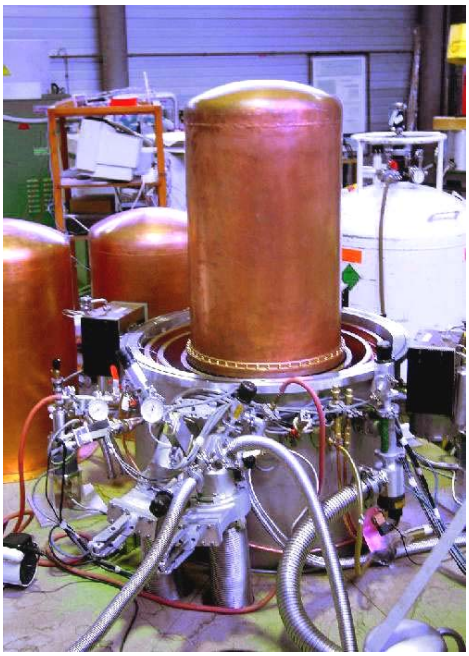


Figure 3 : À gauche, bolomètre équipé d'un anneau de garde. À droite, montage des trois bolomètres de 320g dans le cryostat EDELWEISS-I. Les deux cylindres en plomb sont destinés à protéger les détecteurs.

Afin de maintenir la manipulation à une température de 20mKelvin, celle-ci est contenue dans un cryostat (voir figure 4) fonctionnant à l'azote liquide et différents gaz comme l'hélium.



Vue du cryostat de l'expérience EDELWEISS au Laboratoire souterrain de Modane. Ce cryostat à dilution construit au Drecam/Spec à Saclay permet à l'expérience de fonctionner pendant plusieurs mois en continu à une température de 17 milli kelvins, proche du zéro absolu. Ce cryostat permet d'accueillir trois détecteurs de 320 grammes de germanium.

Figure 4 : Cryostat de l'expérience EDELWEISS au Laboratoire souterrain de Modane

2.3 Les résultats obtenus

La figure 5 (page 16) montre la distribution de Q (facteur de Quenching : rapport ionisation / énergie de recul) en fonction de Er (énergie de recul ou chaleur). La plupart des événements sont à l'intérieur de la bande gamma (en bleu).

L'absence de points dans la zone rouge permet de conclure qu'aucun WIMP n'a interagi dans le détecteur pendant la durée de l'expérience ; tous les points de la figure correspondent à des interactions dues aux rayonnements gamma. Ici, deux points sont présents au sein de cette zone de reculs nucléaires. Ils ne correspondent pas à la détection de WIMPs mais à des erreurs de mesure.

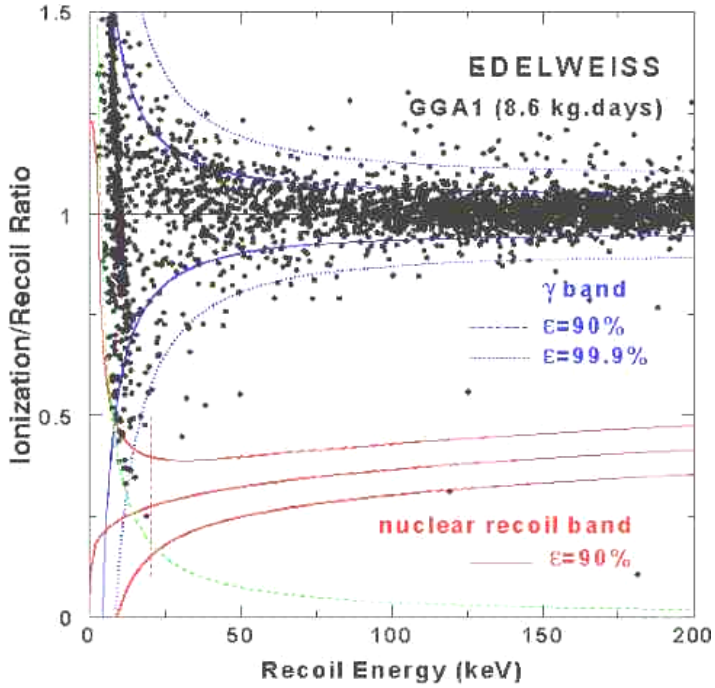


Figure 5 : Facteur de Quenching (rapport de l'ionisation sur l'énergie de recul) en fonction de l'énergie de recul.

La figure suivante est un diagramme d'exclusion tiré des résultats précédents ; il représente la section efficace d'interaction (grandeur liée à la probabilité d'interaction) d'un WIMP avec le détecteur, en fonction de la masse supposée du WIMP. Les limites expérimentales d'ELDELWEISS peuvent être comparées avec celles d'autres expériences (CDMS, DAMA).

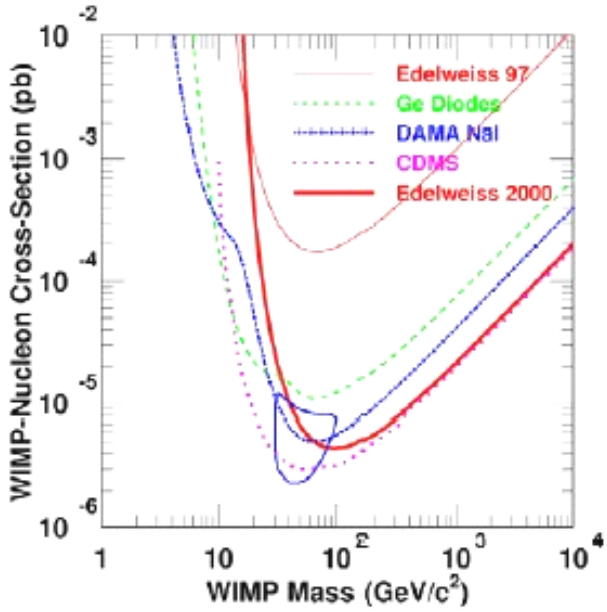


Figure 6 : Courbe d'exclusion, obtenue avec les données de la figure précédente.

Si les WIMPs existent, leur section efficace est nécessairement inférieure aux valeurs données par la courbe rouge en gras. Le contour fermé au centre correspond quant à lui à la zone où l'expérience sino-italienne DAMA (DArk MAtter) pense avoir trouvé quelques WIMPs ; un résultat en désaccord avec celui de l'expérience américaine CDMS (Cryogenic Dark Matter Search) qui exclut une large fraction de cette zone.

Plus une expérience dure longtemps sans que soient mises en évidence d'interactions de WIMPs, et plus la section efficace qu'il est possible de leur attribuer est faible.

A ce stade, nous pouvons penser qu'il suffirait de prolonger l'expérience suffisamment longtemps pour arriver dans la zone rouge de la figure 6 où les modèles supersymétriques prévoient la présence de WIMPs. Mais cela n'est pas si simple. Dans la pratique, il faut augmenter la masse des détecteurs et diminuer le bruit de fond. Ce sera l'objectif de la prochaine expérience, EDELWEISS II, qui utilisera pour cela jusqu'à 120 détecteurs.

Les résultats obtenus après la mise en œuvre d'EDELWEISS II, et ceux d'autres équipes internationales traquant les WIMPs par des techniques différentes, devraient permettre d'apporter des éléments de réponse sur la nature de la matière cachée.

2.4 Vers quoi évolue-t-on ?

EDELWEISS-II :

Cette deuxième étape est basée sur un cryostat disposant de 100 litres de volume utile pour les détecteurs et de la possibilité d'y installer progressivement 120 détecteurs ionisation-chaleur de 320g.

Blindages gammas et neutrons :
Le choix des blindages tient compte de l'expérience acquise auprès de EDELWEISS I et des calculs de simulations. Les blindages gammas doivent protéger les détecteurs du flux mesuré au LSM. Une épaisseur de 20 cm de plomb a été adoptée. De plus un surmoulage interne en plomb archéologique placé au plus près des détecteurs protège ceux-ci du flux interne dû à l'activité du plomb.

Un blindage externe au blindage précédent doit protéger les bolomètres de plusieurs sources neutrons qui, aux profondeurs où se situe le dispositif expérimental, sont d'une part les neutrons produits par les réactions de fission et les réactions alpha, et d'autre part, les neutrons produits par les muons. Les neutrons de faible énergie (<10MeV) sont arrêtés par un blindage de polyéthylène d'épaisseur 50cm. Les neutrons dits muoniques sont créés dans l'environnement (roche, laboratoire) et dans le blindage des détecteurs avec de grandes énergies (> 10MeV). La protection vis-à-vis de ces neutrons est encore à l'étude.

Elle doit faire intervenir un « veto » ou détecteur de muons qui enveloppera le blindage externe.

Bien d'autres modifications sont prévues et en cours de validation ; de la chambre blanche aux choix des différents matériaux composant chaque ensemble et sous ensembles, avec toujours comme but premier de réduire le bruit des différents rayonnements.

Cependant, nous ne traitons, dans le cadre du stage, que le cas des perturbations liées aux sources de neutrons, faisant intervenir le veto-muon.

3 Le veto-muon

3.1 Pourquoi un veto-muon ?

Les neutrons rapides peuvent provenir d'une interaction des muons avec le blindage de plomb. L'objet du veto est de faire une coïncidence entre le veto lui-même et les bolomètres. Le flux de muons et leur énergie ont été mesurés par différentes expériences. La nature du veto sera un assemblage de scintillateurs plastiques dont l'épaisseur de 5 cm est calculée pour que la perte d'énergie muon soit suffisamment grande devant l'énergie des gammas susceptibles de traverser le veto afin d'avoir une bonne discrimination entre muon et gamma.

Les évènements détectés sur les bolomètres pourront ainsi faire l'objet de discrimination dès lors d'une coïncidence avec le système veto-muon.

3.2 Le principe

L'expérience dans son ensemble sera située à l'intérieur d'un coffrage de scintillateurs représenté ci-après. Celui-ci est en cours de développement à l'Université de Karlsruhe en Allemagne.

Le coffrage est constitué de trois parties dont deux sont amovibles pour permettre l'accès à la manipulation.

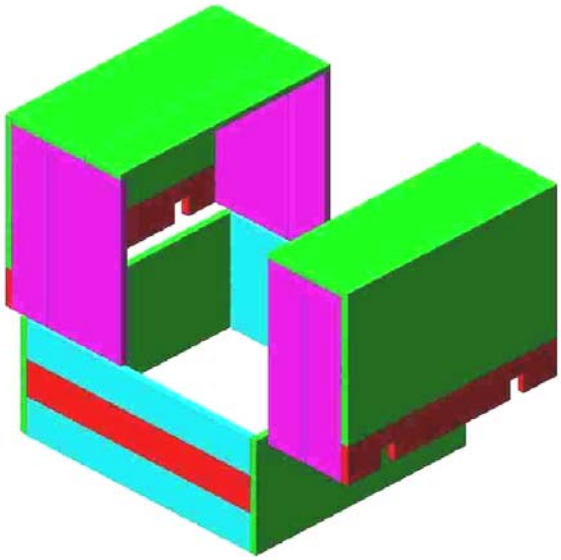


Figure 7 : Coffrage constitué de scintillateurs

Lors du passage d'un muon au travers de l'un des scintillateurs, ce dernier réagit en émettant des photons, qui sont ensuite collectés sur un photo-multiplicateur, dont l'information peut ensuite être traitée par le système électronique veto-muon.

La figure 8 (page 20) représente les différentes sous parties du système véto-muon. Ce dernier dépend du système d'acquisition qui reçoit les données des bolomètres pour les envoyer vers des stations de traitement Macintosh ; ce système d'acquisition permet également d'identifier les différents bolomètres présents dans la manipulation ; et d'autre part de fournir une base de temps au système véto-muon, par l'intermédiaire d'une fibre optique (le choix des fibres optiques a été fait dans le but de créer un isolement galvanique).

Pour cela, nous utilisons une carte nommée ICA (Interface Calculateur d'Acquisition) en cours de développement au CEA, qui envoie une base de temps nécessaire au système véto-muon.

Le système véto-muon quant à lui reçoit les signaux issus des photo-multiplicateurs du coffrage de scintillateurs, pour les transformer en signaux électriques différentiels ECL. Ces signaux, au nombre de sept, représentent l'information de passage d'un muon (signal « latch ») ainsi que l'information géométrique traduisant l'endroit par lequel le muon est passé (numéro de la face du cube matérialisé par le coffrage).

Ces signaux ECL sont ensuite acheminés, par un câble en nappe, sur une carte nommée « time base board », faisant en partie l'objet de ce stage. Cette carte reçoit également la base de temps de la carte ICA. La communication avec la carte « time base board » s'effectue par un bus VME, par l'intermédiaire d'une carte de prototypage VME déjà existante (que nous appelons carte « interface VME »).

Le choix du format électrique ECL a été choisi par l'Université de Karlsruhe, auquel a été confié le développement de la carte d'adaptation des signaux et le traitement des données collectées par l'ensemble du système véto-muon.

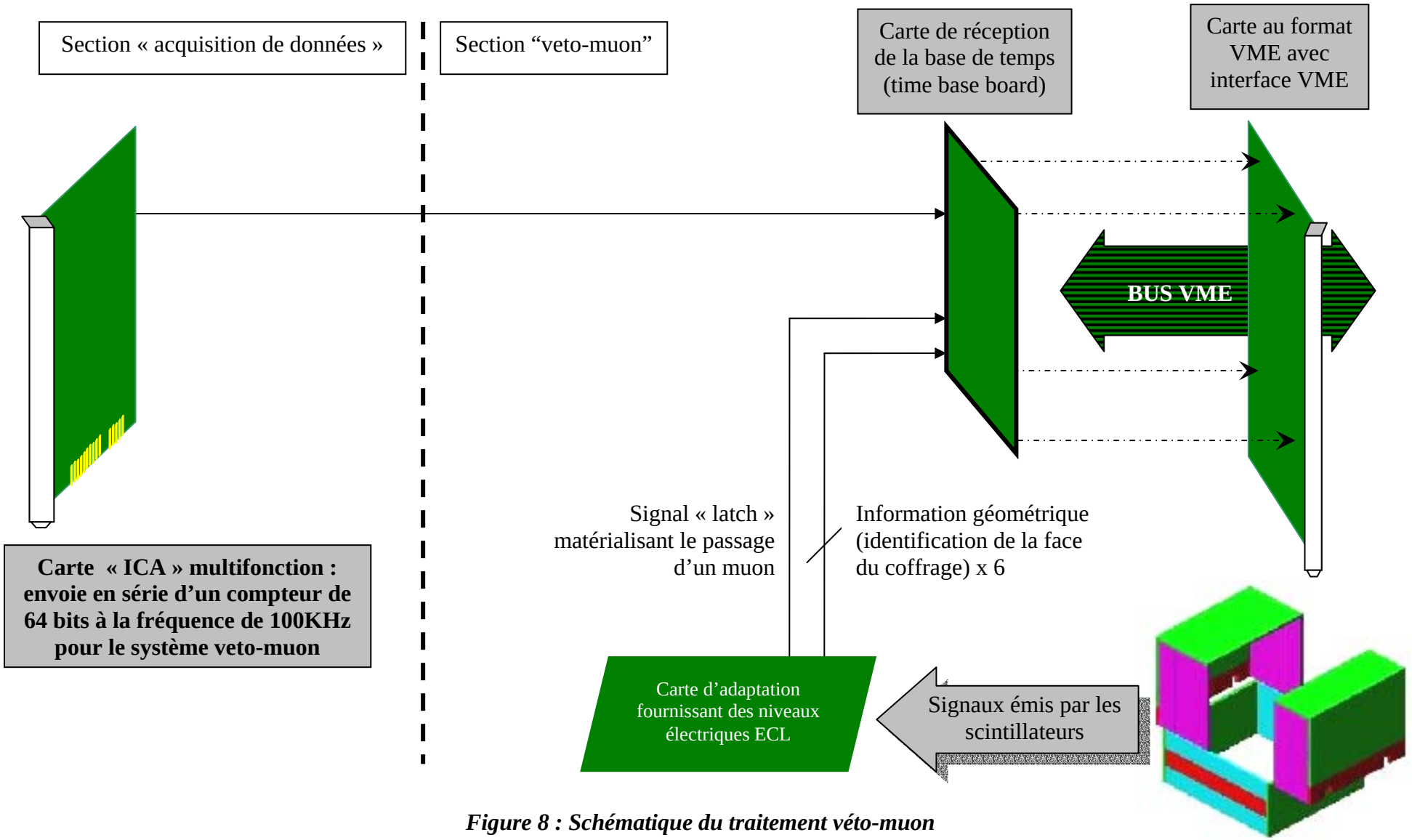


Figure 8 : Schématique du traitement véto-muon

3.3 La carte « time base board »

La première partie du stage a donc été de créer la carte « time base board », basée autour d'un FPGA, de réaliser l'adaptation des signaux ECLs et de s'interfacer avec la carte VME existante.

3.3.1 Généralités

La fonction première de la carte « time base board » est de réceptionner la base de temps de la carte ICA. Cette base de temps sert à dater les évènements muons détectés par l'expérience, avec la même horloge que celle de l'acquisition des données. Ces évènements sont matérialisés par l'information « latch » reçue par la carte « time base board ». Après chaque événement muon, la date de celui-ci est rendue disponible sur le bus VME.

D'autre part, l'information géométrique représentée par les six autres signaux ECL est réceptionnée en vue d'identifier la face du coffrage concernée.

Afin de valider un nouvel enregistrement d'évènement, un signal « enable-latch » peut être transmit à la carte par l'intermédiaire du bus MVE.

De plus, la validation du système véto-muon (réalisée en Allemagne) peut être faite sans la partie acquisition de données, avec la possibilité de choisir (par action sur un interrupteur) une base de temps interne pour émuler celle provenant de la carte ICA. C'est cette même base de temps qui sera implantée par la suite dans la carte ICA.

Le schéma suivant (page 22) résume les fonctionnalités de la carte « time base board ».

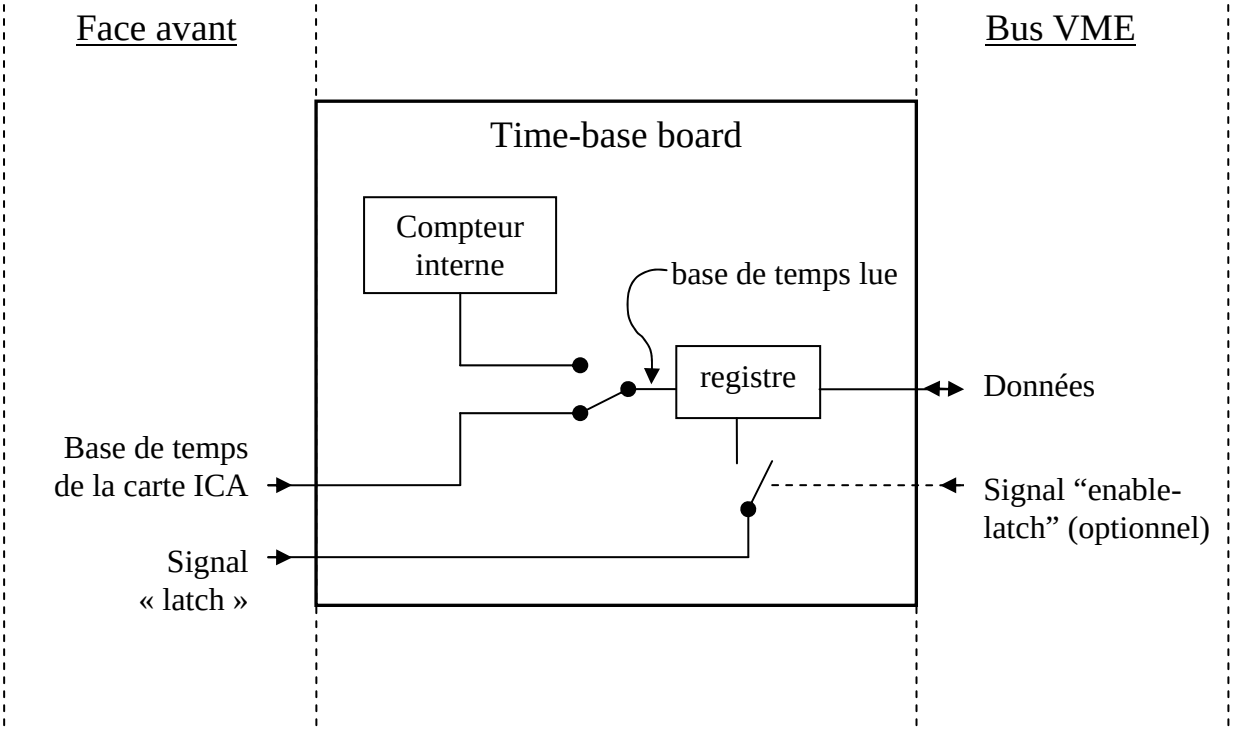


Figure 9 : Schéma de fonctionnement global

3.3.2 Choix matériel

- Pour une raison de temps, la carte a été réalisée sur une carte d’étude 100mm x 160mm de manière à pouvoir l’enficher en mezzanine sur la carte interface VME.

- Les diverses fonctions de la carte sont réalisées grâce au composant programmable Xilinx SPARTAN II xc2s50-5pq208 (176 entrées-sorties, 50000 portes, 384 CLBs), cadencé par un oscillateur à 125MHz. Il est alimenté en 2.5V et possède des entrées-sorties 5V tolérantes, rendant possible l’adaptation avec l’interface VME.

- L’oscillateur est un IQXO-331 de chez C-MAC et est donné pour une fréquence de 125.0MHz.

- Le FPGA est programmé, pour des raisons pratiques de reconfiguration, par une PROM Xilinx XC18V01 reprogrammable par liaison boundary-scan.

- La carte nécessite d’avoir les alimentations 2.5V pour le FPGA et 3.3V pour la PROM ainsi que 5V et -5V pour le récepteur optique. Ces tensions sont obtenues grâce à des régulateurs qui utilisent les tensions d’alimentation disponibles sur le bus VME (5V, 12V et -12V).

- Le récepteur optique est de type ST (HFBR 2416T d'Agilent-Technologies). La chaîne de réception est constituée d'un amplificateur et de deux comparateurs de façon à traduire l'information optique en grandeur électrique TTL.

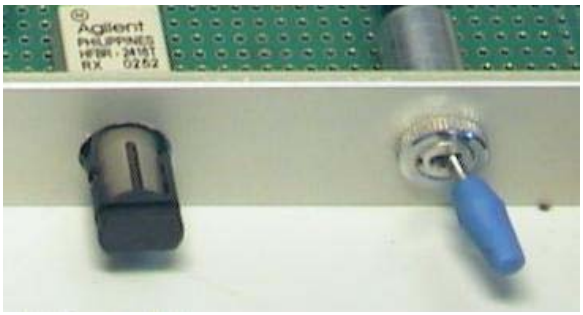


Figure 10 : A gauche le récepteur fibre optique, à droite l'interrupteur de sélection de la base de temps (interne ou provenant de la carte ICA)

- Les signaux ECLs différentiels proviennent d'un câble en nappe, la réception sur la carte se fait via un connecteur de type 2x8 broches représenté ci-dessous. Les sept signaux ECLs ne pouvant être traités directement par le FPGA, nous utilisons sept convertisseurs ECL-TTL de type MC100ELT de Motorola.

Figure 11 : Connecteur acceptant les sept signaux ECLs différentiels

Remarque : Le schéma de câblage de l'ensemble de la carte est donné ci-après (figure 12, page 24). Par manque de place, certains composants ont été placés sur la carte interface VME (figure 13, page 25).

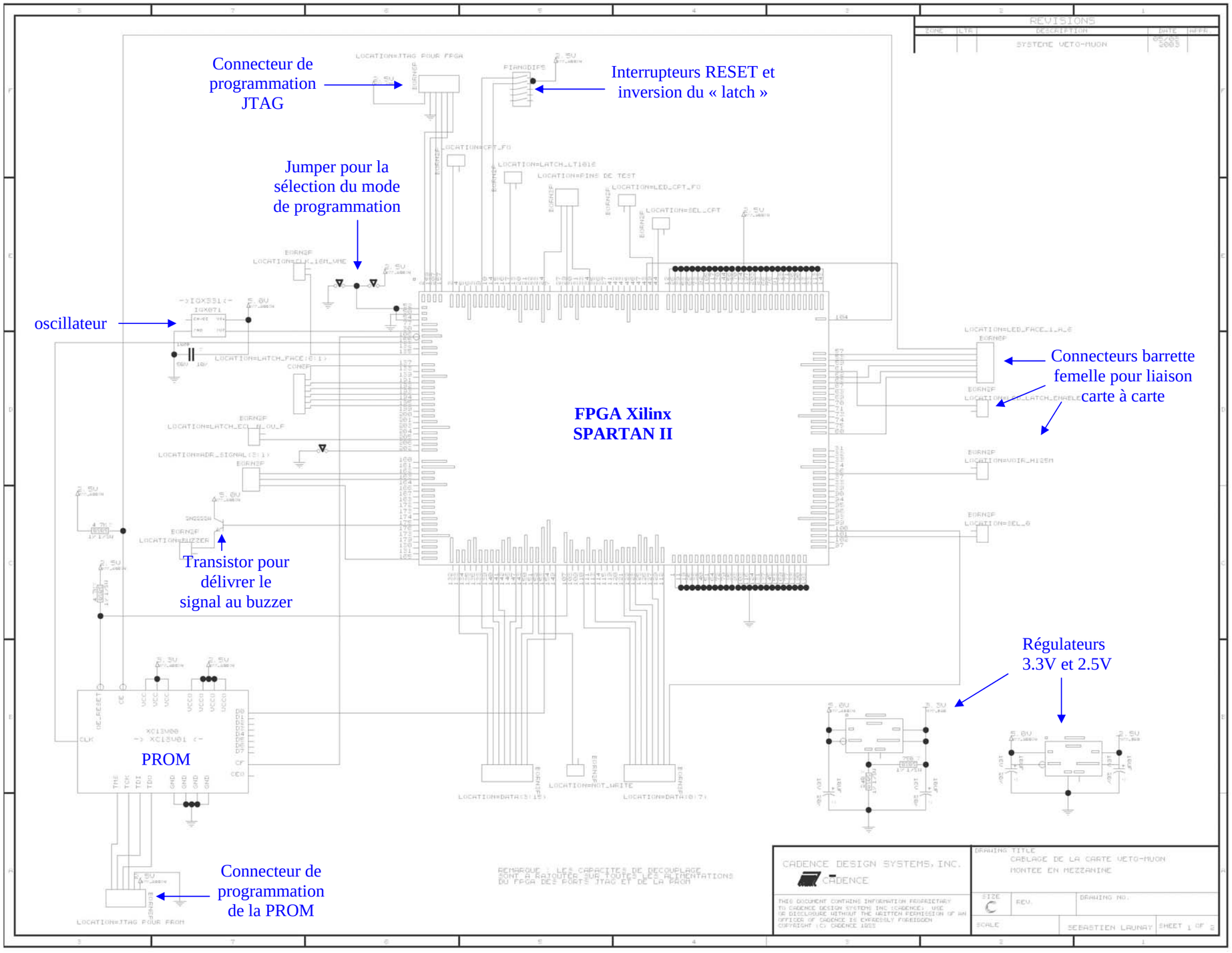


Figure 12 : Câblage réalisé sur la carte "time base board"

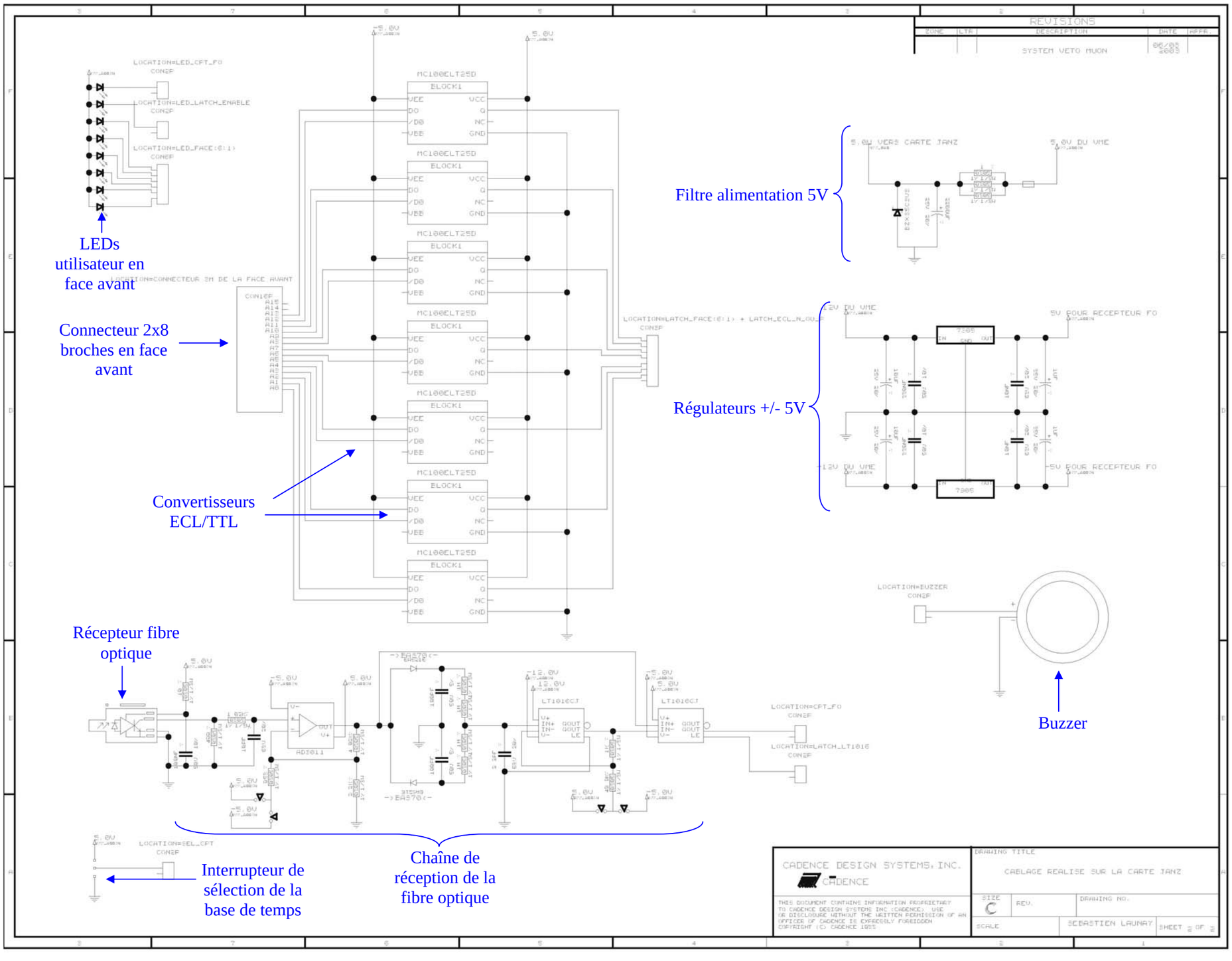


Figure 13 : Câblage réalisée sur la carte interface VME

3.3.3 Principe de fonctionnement

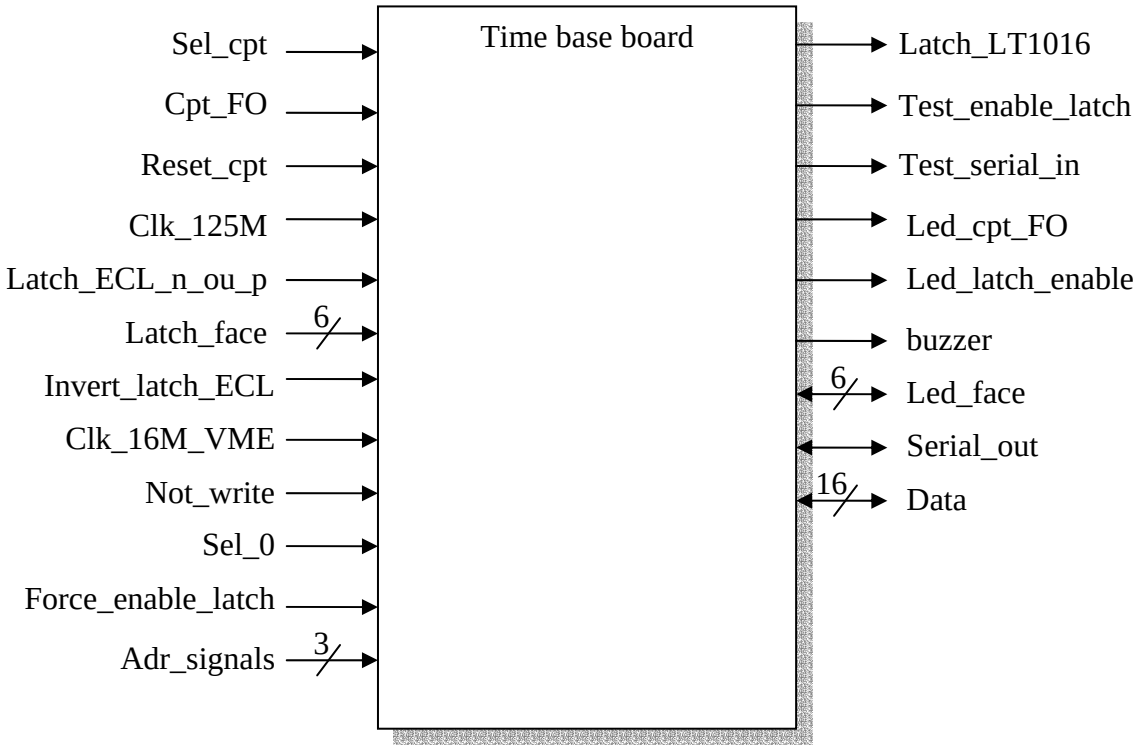


Figure 14 : Schéma bloc de la programmation VHDL

Description des différentes entrées/sorties :

Les entrées :

Sel_cpt	Permet la sélection entre la base de temps externe (par fibre optique) ou interne, via l'interrupteur de face avant.
Cpt_FO	Base de temps externe, en sortie du comparateur LT1016 de la chaîne de réception fibre optique.
Reset_cpt	Permet une remise à zéro de la base de temps interne, via le DIP switch placé sur la carte « time base board ».
Clk_125M	Horloge 125MHz en sortie de l'oscillateur.
Latch_ECL_n_ou_p	Signal « latch » en sortie du convertisseur ECL/TTL, provenant du connecteur 2x8 broches de la face avant.
Latch_face (6:1)	Information géométrique concernant le numéro de la face du scintillateur.
Invert_latch_ECL	Permet d'inverser la logique du signal latch_ECL.
Clk_16M_VME	Horloge 16MHz provenant du VME, nécessaire au protocole de communication.
Not_write	Signal VME d'écriture/lecture
Sel_0	Signal VME de présélection des adresses.
Force_enable_latch	Permet de rendre le signal « latch » optionnel.
Adr_signals (3:1)	Signaux VME permettant la réception du « latch » ou l'émission sur le bus VME de la base de temps enregistrée.

Les sorties :

Latch_LT1016	Permet de désactiver le comparateur de la chaîne de réception fibre optique.
Test_enable_latch	Signal « enable_latch » disponible sur un plot de test de la carte « time base board ».
Test_serial_in	Idem pour la base de temps envoyée en série.
Led_cpt_FO	Permet de commander la LED en fonction de la sélection de la base de temps.
Led_latch_enable	Permet de commander la LED lorsque le signal « enable_latch » est reçu.
buzzer	Envoie un signal modulé au buzzer, lors de l'arrivée d'un muon (c'est-à-dire du signal « latch_ECL_n_ou_p »).

Les bidirectionnels :

Led_face (6:1)	Permet l'affichage du numéro de face concerné (matérialise les signaux « latch_face »).
Serial_out	Base de temps interne.
Data (16:1)	Signaux VME pour l'envoi de la base de temps enregistrée ou la réception du signal « enable_latch ».

Les signaux « led_face » sont de type bidirectionnel (et non pas uniquement des sorties) pour permettre, en fonction de leurs états, de commandes une temporisation décrite dans le paragraphe sur les autres fonctionnalités.

Fonctionnement de la carte « time base board » :

- La base de temps :

La base de temps (envoyée en série sur la fibre optique ou interne à la carte), servant à dater les événements muons, est réalisée grâce à un compteur 64 bits s'incrémentant toutes les 10µs. Les bits sont envoyés en série à 16Mbits/s.

La trame comportant le compteur possède un bit de start (à 0) et de stop (à 1) permettant lors de la réception de cette trame, de décoder la base de temps utile. Le reste du temps, la trame est maintenue à l'état 1 :

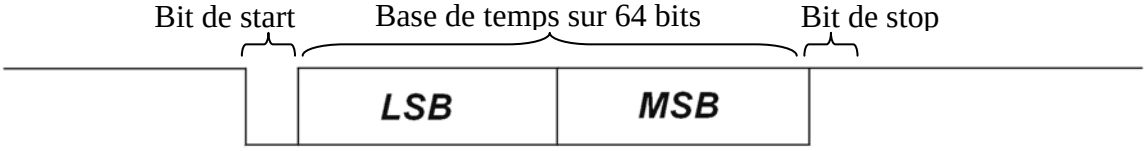


Figure 15 : Constitution de trame comportant la base de temps.

Cette configuration nous permet de compter durant plusieurs millions d'années, ce qui ne présente pas d'intérêt primordial dans le cadre de l'expérience EDELWEISS II.

Afin de limiter le nombre de registres et donc de CLBs (Configurable Logic Block) utilisés, nous choisissons de n'incrémenter que sur les 48 premiers bits du compteur interne ; les 16 derniers étant maintenus constamment à zéro dans un registre séparé. La nouvelle trame possède donc une taille de 50 bits (compteur 48 bits + bit de start + bit de stop).

Ceci nous permet cependant de compter jusqu'à 281 474 976 710 655 ; ce qui rend une utilisation possible sur 89 ans consécutifs, à la fréquence de 100KHz.

- La permission de dater un événement :

Afin de pouvoir enregistrer la date (c'est-à-dire la valeur de la base de temps) d'un événement muon, l'utilisateur doit envoyer à la carte « time base board » un signal de permission « enable_latch » pour confirmer qu'il est en mesure de traiter l'événement. Cependant, cette permission peut être rendue optionnelle grâce à un jumper place sur la carte (voir la figure 16).

L'envoi du signal « enable_latch » se fait via le bus VME. Il faut, selon le protocole VME, que l'ensemble carte « time base board » et interface VME soit sélectionné (signal « sel_0 » à 0) et en mode écriture (signal « not_write » à 0). D'autre part, nous avons fixé la convention de devoir sélectionner l'adresse « adr_signal(3) » et de positionner la donnée « data(0) » à 1 pour mettre à 1 le signal « enable_latch ».

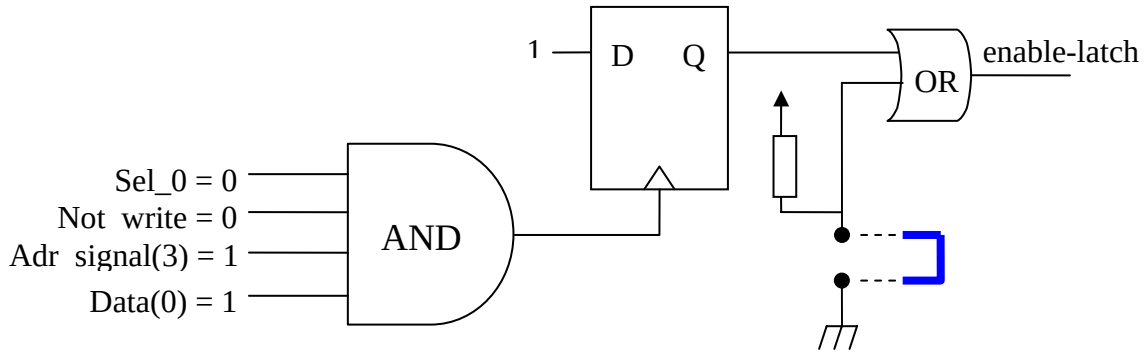


Figure 16 : Logique utilisée pour mettre le signal « enable-latch » à 1.

Remarque : Si un ou plusieurs événements muon apparaissent alors que le signal « enable_latch » est positionné à 1, l'événement muon enregistré sera le dernier. Les autres ne seront pas considérés.

La remise à zéro du signal « enable_latch » se fait uniquement lorsque le signal « latch_ECL_n_ou_p » (c'est-à-dire lors d'un événement muon) devient actif.

Le signal « enable_latch » peut être rendu optionnel en enlevant le jumper bleu, ce qui a pour effet de forcer à 1 ce dernier. Cela a pour conséquence de rafraîchir la date enregistrée à chaque événement muon, ce qui peut poser problème dans le cas où les événements sont rapprochés et si l'ensemble du système véto-muon n'a pas le temps de traiter tous ces événements.

- L'enregistrement de la date d'un événement :

Dès lors que le signal « enable_latch » est actif, la date d'un événement muon (matérialisée par le signal « latch_ECL_n_ou_p ») peut être enregistrée à tout instant.

Pour cela, la base de temps reçue est enregistrée dans un registre de 48 bits au moment de l'événement. Par ailleurs, si la base de temps est en train de s'incrémenter au moment de l'événement, l'enregistrement de sa valeur ne se fera que lorsque cette dernière sera stable.

Les figures suivantes illustrent ces propos. Elles représentent les chronogrammes du programme VHDL relevés sur un oscilloscope, à partir d'une carte de test. La figure 17 montre la base de temps (en vert) envoyée en série, s'incrémentant toutes les 10µs. La largeur du compteur fait 48 bits ; chaque bit étant envoyé à 16MHz. En dehors du compteur, la valeur de la trame est maintenue à 1. Les poids faibles du compteur sont situés à gauche. Nous pouvons distinguer l'incrémentation de la base de temps (modification des bits de poids faible), entre la valeur lue à gauche et à droite.

Le signal jaune correspond à un événement muon sur « latch_ECL_n_ou_p » (actif sur le front descendant). Le signal bleu quant à lui représente l'enregistrement de la base de temps. Nous pouvons voir que cet enregistrement a bien été mis à jour suite à l'événement.

Pour plus de précision, les valeurs notées 1 et 2 ont été agrandies sur les figures 18 et 19, pour être comparées ; il apparaît nettement que les valeurs sont identiques ce qui traduit que la base de temps enregistrée est la bonne.

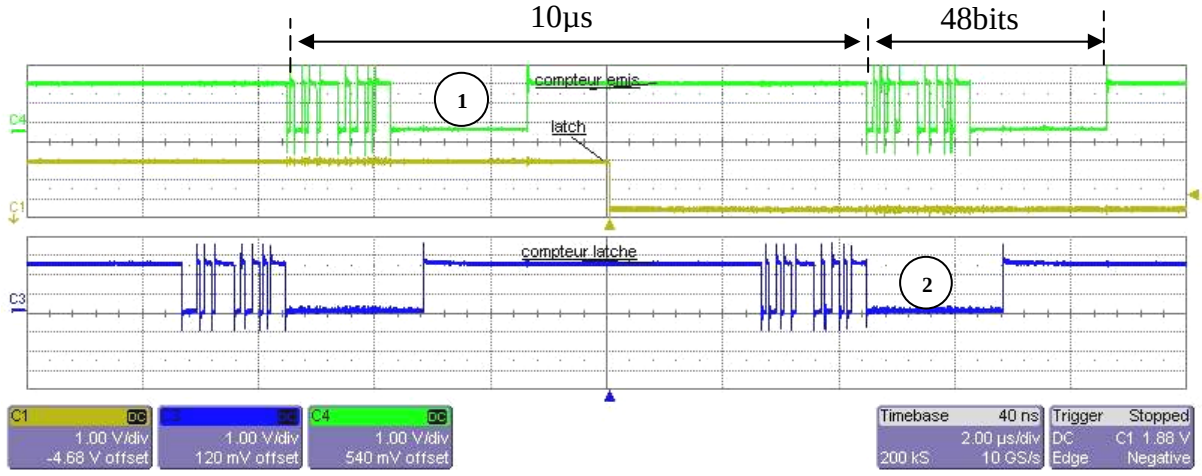


Figure 17 : Chronogrammes de la base de temps et de son enregistrement associé

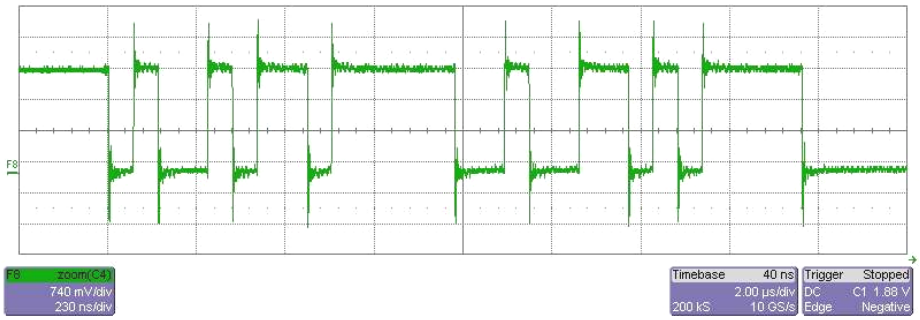


Figure 18 : Agrandissement de la valeur de la base de temps reçue

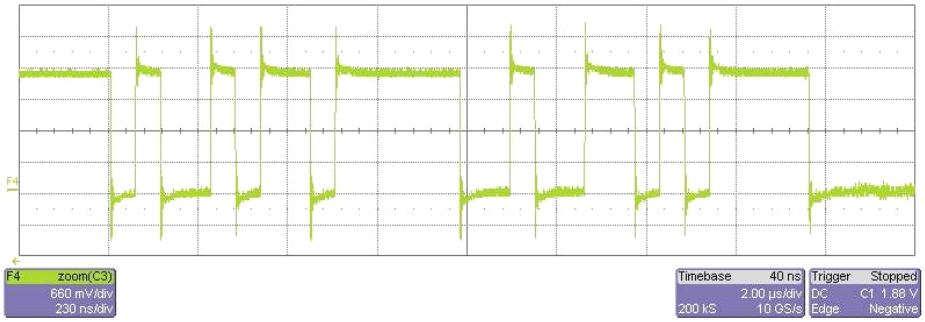


Figure 19 : Agrandissement de la valeur de la base de temps enregistrée

- Lecture de la base de temps enregistrée

Après avoir enregistré la base de temps, suite à un événement muon, sa valeur est rendue disponible sur le bus VME.
Le registre où se situe cette valeur peut être lu à tout moment. Il suffit pour cela de d'activer le signal « sel_0 » (actif à l'état 0) comme précédemment et de spécifier le mode « lecture » avec le signal « not_write » à 1.

Etant donné que le compteur a une taille totale de 64 bits, et que le bus de données est limité à 16 bits, les données sont présentées sur le bus VME via les adresses « adr_signals(1) » et « adr_signals(2) » (l'adresse « adr_signals(3) » étant maintenue à 0), selon le tableau suivant :

A3	A2	A1	Compteur de la base de temps [63 : 0]	Mode
0	0	0	"0x0000"	lecture uniquement
0	0	1	compteur [47 : 32]	
0	1	0	compteur [31 : 16]	
0	1	1	compteur [15 : 0]	
1	X	X	XXX	écriture uniquement

Tableau 1 : Affectation, sur le bus VME, de la base de temps enregistrée en fonction de l'adressage.

Remarque :
Si plusieurs événements muon sont reçus avant qu'ils aient pu être lus, seul le premier de ceux-ci sera enregistré et par conséquent lisible. Cela n'est valable que dans l'utilisation du signal « enable_latch » ; si ce dernier est rendu optionnel, l'événement pris en compte sera le dernier reçu.

D'autre part, aucun événement muon ne peut être enregistré tant que le mode lecture est activé.

3.3.4 Autres fonctionnalités et vue d'ensemble

Le cahier de charges de l'expérience EDELWEISS II étant en constante évolution, des fonctionnalités secondaires ont été mises en place sur l'ensemble des cartes « time base board » et interface VME.

- La figure suivante représente un double interrupteur permettant (pour effectuer des tests) de mettre à zéro la base de temps interne (interrupteur 1) et d'inverser le signal « latch_ECL_n_ou_p » (interrupteur 2).

  Les trois broches de test (au centre de la figure) permettent la vérification des signaux correspondant respectivement (de gauche à droite) à la base de temps interne, la base de temps lue et le signal « enable-latch ».

  Le jumper, à gauche, permet d'effectuer une programmation série du FPGA en mode maître ou boundary-scan.

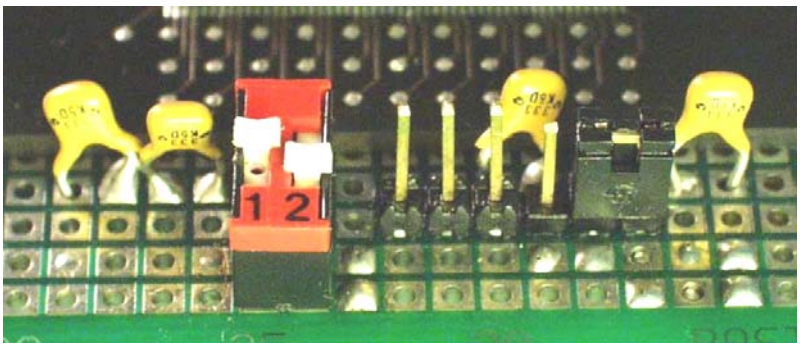


Figure 20 : De gauche à droite : interrupteurs et broches de test, sélection du mode de programmation.

- Le circuit de réception optique étant toujours alimenté, ce dernier décode constamment la valeur présente sur la fibre optique, même lorsque celle-ci n'est pas sélectionnée.
  Afin d'éviter cette activité inutile lorsque la base de temps interne est sélectionnée, nous utilisons l'entrée « enable » disponible sur le comparateur LT1016 qui permet de mettre en haute impédance la sortie du montage de réception fibre optique.
  Ce signal est commandé par l'interrupteur de sélection de la base de temps, puis est inversé dans le FPGA avant le comparateur LT1016.

- A la demande de l'Université de Karlsruhe en Allemagne (qui réalise le système véto-muon), le signal « enable_latch » peut être rendu optionnel en fonction des besoins des différents tests et de l'expérience. Pour cela, un simple jumper placé sur la carte « time base board », permet d'enregistrer la date d'arrivée d'un évènement muon sans que le signal « enable latch » ait été envoyé. Dans ce cas, et si plusieurs muons sont détectés successivement, seul le dernier évènement pourra être lu, étant donné que le registre est rafraîchi à chaque signal « latch ».

- La face avant de la carte interface VME possède huit LEDs utilisateurs que nous avons affectées comme suite :

  La LED U0 permet de confirmer la sélection de la base de temps en cours : la LED s'allume dans le cas où la base de temps interne est sélectionnée et clignote à la période d'une seconde lorsque la base de temps provient de la carte ICA via la fibre optique.

  La LED U1 traduit l'état du signal « enable_latch » (allumée lorsque le signal est à l'état haut ou lorsque celui-ci est rendu optionnel).

Les LEDs U2 à U7 représentent l'information géométrique de la face du cube composé par les scintillateurs. La LED concernée est activée en même temps que la réception du signal « latch » et s'éteint à l'arrivée du signal « enable_latch ». Si ce dernier est rendu optionnel, la LED est désactivée après une temporisation de cinq secondes.

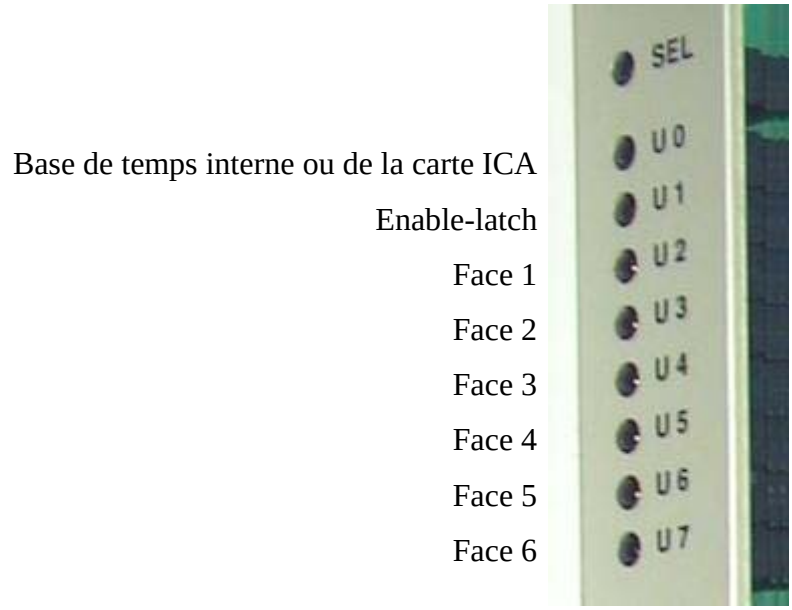


Figure 21 : Affectation des LEDs utilisateur de la face avant de la carte interface VME

- De façon à obtenir une confirmation de la réception d'un événement muon par le signal « latch_ECL_n_ou_p », un son d'une seconde est produit sur un buzzer, situé sur la carte interface VME.

  Pour cela, le FPGA réceptionne le signal « latch_ECL_n_ou_p » puis envoie, par l'intermédiaire d'un transistor, un signal de 50Hz avec un rapport cyclique de 50%. Tous ces paramètres sont facilement modifiables dans le code VHDL, afin de pouvoir agir sur la durée et la modulation du son émis.

Ci-dessous, la vue de dessus de la carte « time base board » avec au centre le FPGA Xilinx SPARTAN II et ses capacités de découplage d'alimentation. Les barrettes femelles bleues permettent la connexion avec la carte interface VME. Le câblage du FPGA a été réalisé avec des fils de wrapping soudés (visibles sur la figure 23).

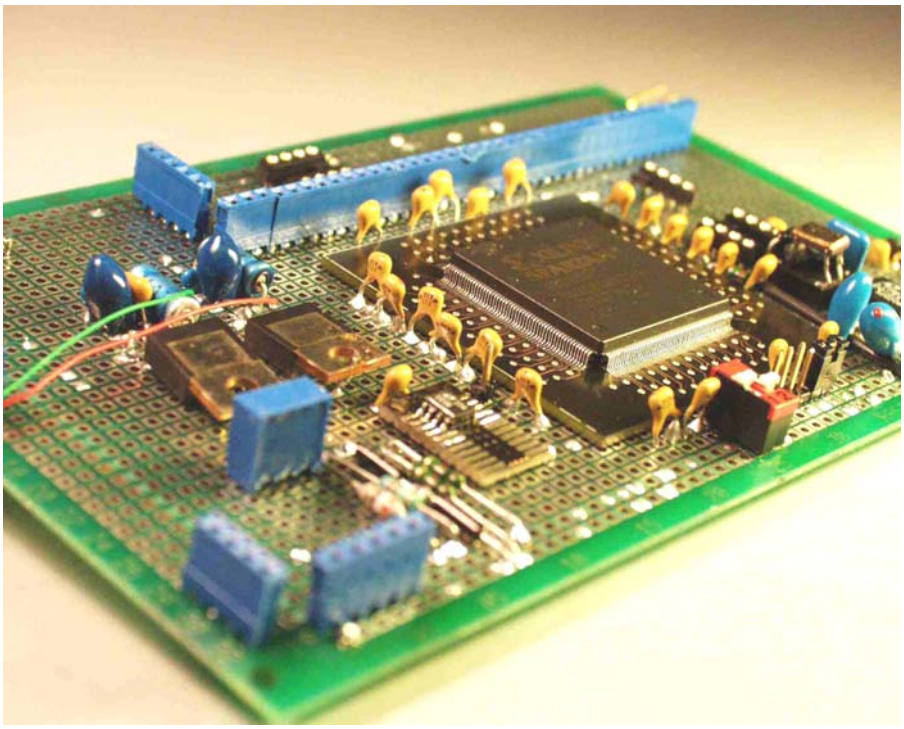


Figure 22 : Carte "time base board" seule.

Ci-dessous, l'ensemble « time base board » (vue de dessous) monté en mezzanine sur la carte interface VME. Sur la face avant, de gauche à droite, se trouvent les LEDs utilisateurs, le connecteur 2x8 broches pour la réception des signaux ECLs différentiels, le récepteur fibre optique et l'interrupteur de sélection de la base de temps.

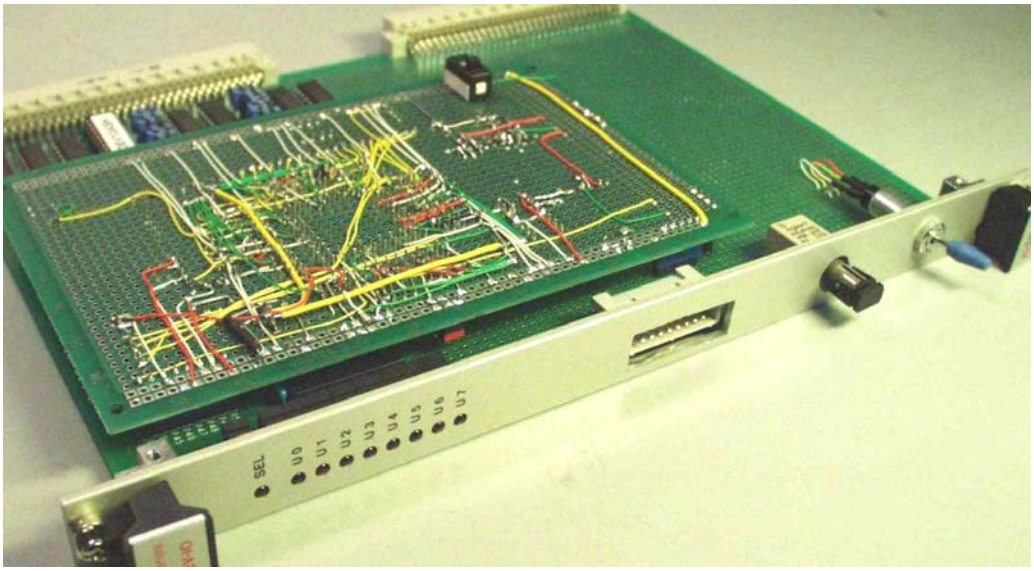


Figure 23 : Ensemble carte "time base board" et carte interface VME.

3.3.5 Description du code VHDL

La totalité du code VHDL est disponible en annexe 1 (page 61).

Nous ne détaillons que les points principaux du code afin de mieux comprendre, dans son ensemble, la logique choisie pour parvenir à dater un événement muon, à partir de la base de temps reçue sur fibre optique ou à partir de la base de temps interne qui doit elle-même être explicitée.

Remarque :
Pour des raisons de temps de traitement interne au FPGA, la fréquence de 125MHz issue de l'oscillateur est divisée par deux à l'entrée du FPGA. Cela à pour conséquence de diviser également par deux les compteurs des différentes fréquences qui en dépendent.

- **Génération de la base de temps interne :**

Cette partie du code s'inscrit à l'intérieur du process lié à l'horloge 125MHz.
Afin d'incrémenter, toutes les 10µs, le compteur 48 bits représentant la base de temps, nous avons besoin d'une horloge à 100KHz. Nous réalisons cette dernière à l'aide du compteur :

```
-------- generation du signal 100KHz ----------
i <= i + 1 ;
if i = 624 then
        i <= 0 ;
end if;
```

La trame envoyée en série comporte le compteur 48 bits ainsi que les bits de start et de stop. Toutes les 10µs, cette trame prend donc une nouvelle valeur en incrémentant le compteur :

```
trame <= '1' & (trame(48 downto 1) + un) & '0' ;
```

Pour être envoyée en série, la base de temps doit être « shiftée ». Nous utilisons pour cela une deuxième trame (trame_I) qui après avoir été mise à jour (*trame_I <= trame*) subit un décalage à la fréquence de 16MHz (correspondant à un envoi à 16bits/s). Lorsque la trame n'envoie pas l'information du compteur, celle-ci reste positionnée à 1 avant d'envoyer la nouvelle valeur du compteur, 10µs plus tard. Le bit entrant dans le décalage est donc un 1 :

```
trame_I <= '1' & trame_I(49 downto 1) ;
```

Afin d'envoyer la base de temps en série, il suffit de récupérer à tout moment le bit zéro de cette deuxième trame. Cette ligne de code se situe donc en dehors des process, et sera mise à jour à la fréquence 16MHz (tout comme trame_I) :

```
serial_out <= trame_I(0) ;
```

- **Réception de la trame comportant la base de temps :**

Que ce soit pour décoder la base de temps reçue en série sur la fibre optique ou provenant du compteur interne, la réception se fait de la même façon : il faut tout d'abord trouver le bit de start pour commencer le décodage au bon endroit de la trame.

Pour cela, un compteur « cpt_a_16MHz » sert à pointer sur chaque bit (la trame étant envoyée à 16Mbits/s) ; ce qui permet alors de compter le nombre de 1 situé dans la trame, avec le compteur « cpt_bit_a_1 ». En effet, si plus de 48 bits consécutifs sont à 1, cela veut dire que l'on se situe entre deux valeurs de compteur :

Figure 24 : Recherche du bit de start.

La fois suivante, où le compteur « cpt_a_16MHz » pointe sur un bit à zéro, traduit l'apparition du bit de start.

```
-------- on compte jusqu'a 49 bits consécutifs a 1 --------
if cpt_bit_a_1 = 49 then
        cpt_bit_a_1 <= 0;
end if;
-- si 49 bits consécutifs a 1 alors on attend le bit de start
if cpt_a_16MHz = 3 and cpt_bit_a_1 = 48 then
        wait_start <= '1' ;
end if;
```

Après avoir trouvé ce bit de start, il faut continuer à pointer sur les bits de la trame, et relever l'état de chaque bit. Cette opération devant être réalisée 48 fois, le compteur « cpt_des_64_bits » permet la capture de tout les bits de la base de temps. L'état de chaque bit est enregistré au fur et à mesure dans un registre à décalage de 48bits (« trame_decodee »).

D'autre part, il faut tenir compte des temps de montée, de descente, du « jitter » et des éventuels bruits de changement d'état des bits. Pour cela, le passage à 1 du compteur « cpt_a_16MHz » servira de condition à la capture de bit, étant donné le positionnement central de cette valeur par rapport au « temps bit ». Cela nous confère une marge de sécurité correspondant à un « jitter » maximal de 31.25ns, ce qui est largement suffisant (le « jitter » possible étant de l'ordre de la nanoseconde).

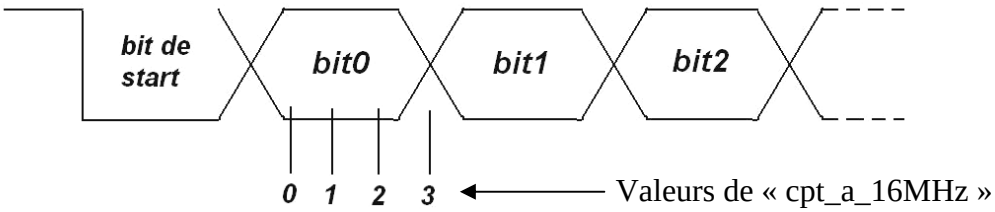


Figure 25 : Décodage de la trame comportant la base de temps.

```
---------- capture du bit de la trame ----------
if cpt_a_16MHz = 1 then
        cpt_des_64bits <= cpt_des_64bits + 1;
        trame_decodee <= serial_in & trame_decodee(47 downto 1);
end if;

------ si l'on a decodé les 50 bits, on arrete ------
if cpt_des_64bits = 49 then
        enable_decode <= '0';
        cpt_des_64bits <= 0;
end if;
```

- **Enregistrement de la base de temps suite à un événement muon :**

Les opérations de génération et réception de la base de temps sont faites constamment, ce qui implique que la base de temps reçue « trame_decodee » est remise à jour toutes les 10µs.

Afin de sauvegarder la date d'un événement muon, nous avons besoin d'un autre registre de 48 bits « cpt_latche » qui sera mis à jour uniquement lorsqu'un événement est enregistré.

Par ailleurs, cette mise à jour ne doit pas être effectuée lorsque la « trame_decode » est en train de se remplir. Ceci implique que le signal « latch_ECL_n_ou_p », représentatif d'un événement, doit être mémorisé (dans le signal « upgrade_cpt_latch ») dans le cas où la base de temps n'a pas finie d'être reçue.
Le signal « enable_decode » quant à lui représente cette mise à jour de la « trame_decode » :

```
 ---------- mise à jours du compteur latché ----------
if enable_decode = '0' and upgrade_cpt_latche = '1' then
        upgrade_cpt_latche <= '0' ;
        cpt_latche <= trame_decodee ;
end if;
```

- **Lecture, par le bus VME, de la base de temps enregistrée.**

Le protocole de communication VME réagit en fonction d'une horloge à 16MHz. Non récupérons cette horloge « clk_16M_VME » afin de réaliser un nouveau process dédié à s'interfacer avec le bus VME.

Comme nous l'avons vu précédemment, la lecture de la base de temps enregistrée après un événement muon, se fait lorsque les signaux de d'écriture « not_write » et de sélection « sel_0 » sont respectivement à 1 et 0. Dans ce cas, notre carte « time base board » met à disposition sur son bus des données « data » les valeurs des bits composant la base de temps enregistrée. Ces bits étant sélectionnés en fonction de l'adresse envoyée sur le bus VME, selon le tableau 1.

```
---------- le VME lit les valeurs du compteur latché ----------
if not_write = '1' and sel_0 = '0' then          -- le bus VME est en lecture
        case adr_signals is
                when "000" =>
                        data <=  "0000000000000000" ;      --en vue d'un compteur 64 bits
                when "001" =>
                        data <= cpt_latche(47 downto 32);
                when "010" =>
                        data <= cpt_latche(31 downto 16);
                when "011" =>
                        data <= cpt_latche(15 downto 0);
                when others =>
                        null ;
        end case;
end if;
```

Par ailleurs, le compteur de 48 bits pourra, selon les besoins futurs, repasser à 64bits ; les 16 derniers bits étant disponibles pour les adresses « 000 ».

- **Réception du signal « enable_latch », provenant du bus VME :**

Cette fois ci, le bus VME doit, d'une part, écrire vers la carte « time base board » et d'autre part, activer le signal « enable_latch ». Notre carte doit donc attendre que les conditions sur les signaux « not_write », « sel_0 », « adr_signals(3) » et « data(0) » soit satisfaites :

```
---------------- réception d'un enable_latch provenant du VME ----
if not_write = '0' and sel_0 = '0' and adr_signals(3) = '1' then   -- le bus VME est en écriture
        data <= "ZZZZZZZZZZZZZZZZ" ;   -- on attend une data
        if data(0) = '1' then
                enable_latch <= '1';
        end if;
end if;
```

3.3.6 Validation du code

Le code VHDL a été saisi sous le logiciel Xilinx ISE.
Afin de valider ce code, le logiciel ModelSim a permis d'effectuer les simulations utiles, qui n'apparaissent pas ici pour une question de lisibilité.

Après simulations, des essais ont été réalisés sur des kits de développement « Memec Design », comportant un FPGA Xilinx Virtex-II PRO XC2VP4-FG-456 (figure 26, page 40). Les entrées du système (interface VME, événements muon) ont été simulées grâce aux interrupteurs (boutons poussoirs et DIP switch placés sur le kit de développement). Les sorties ont été visualisées grâce aux LEDs présentent sur le kit et grâce aux connecteurs « ACE » permettant d'afficher les signaux sur un oscilloscope (figures 17, 18, 19, pages 29 et 30).

Afin de valider parfaitement la réception de la base de temps externe du point de vue du code VHDL, un deuxième kit de développement a été utilisé pour envoyer cette base de temps.

Dans un deuxième temps, après avoir réalisé la carte « time base board », l'un des kits de développement a permis de tester l'ensemble de la chaîne de réception de la base de temps : le kit envoie la base de temps en série sur un émetteur optique, qui via une fibre optique transmet le signal à l'ensemble des cartes « time base board » et « interface VME ». Le signal est reçu par la chaîne de réception optique, avant d'être transmis au FPGA qui traite l'information comme décrit dans la section précédente.

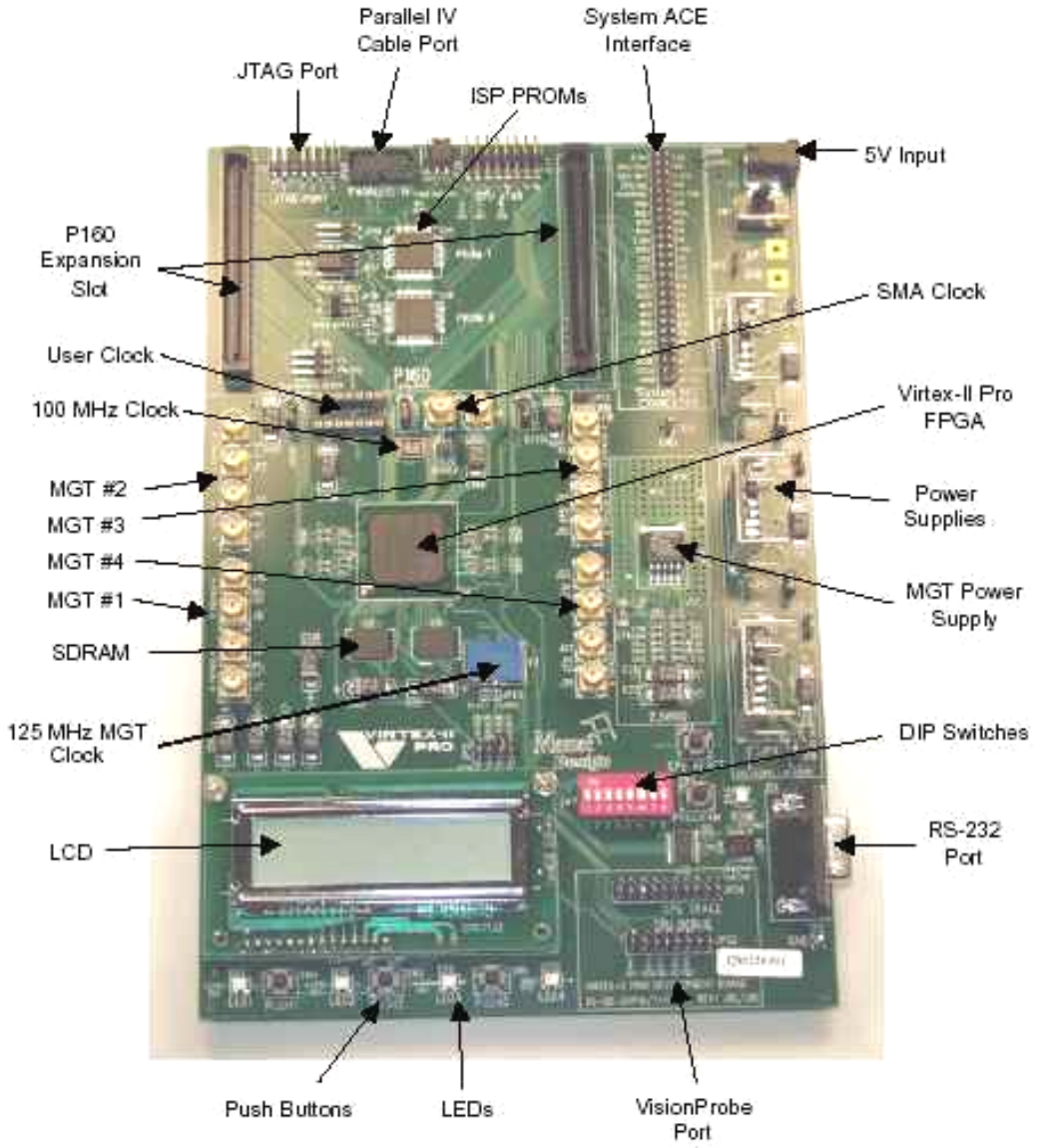


Figure 26 : Kit de développement avec FPGA Xilinx Virtex II PRO.

3.4 Difficultés rencontrés

La première version de la carte « time base board » a été réalisée autour d'un FPGA Xilinx spartan xcs10-plcc-84, qui présentait l'avantage d'être sur support enfichable (facilités de câblage) et qui possédait une alimentation et des entrées/sorties 5V (identiques à l'interface VME). Cependant, ayant mal jugé de la taille occupé par le programme VHDL, ce dernier a dû être modifié (en particulier : passage de la base de temps sur 48 bits au lieu de 64) pour ne pas dépasser le nombre de CLBs disponibles dans le FPGA.

D'autre part, après avoir mis sous tension l'ensemble des cartes « time base board » et « interface VME », un court-circuit lié au FPGA nous a amené à devoir changer ce dernier.

Cependant, la série Spartan n'étant plus disponible dans des délais appropriés, nous avons dû faire le choix de réaliser un second câblage, autour d'un Xilinx SPARTAN II XC2S50-5PQ208, dont les entrées/sorties sont 5V tolérantes.

Après test de l'ensemble du système (carte « time base board » et « interface VME » dans un châssis VME), nous avons constaté des disfonctionnements liés aux alimentations : les tensions 5V et 2.5V sont fabriquées à partir de la tension 5V, disponible sur le bus VME ; or, lorsque les cartes sont débranchées, le 5V du bus VME présente des variations (chronogramme en marron) mais qui restent raisonnables (chute de 25% de la tension pendant environ 300µs). Cependant, dès l'instant que la carte « interface VME » est branchée, cela induit des chutes de tensions proches de 0V à plusieurs reprises (chronogramme en rouge), ce qui se répercute sur les tensions du FPGA (chronogrammes en bleu).

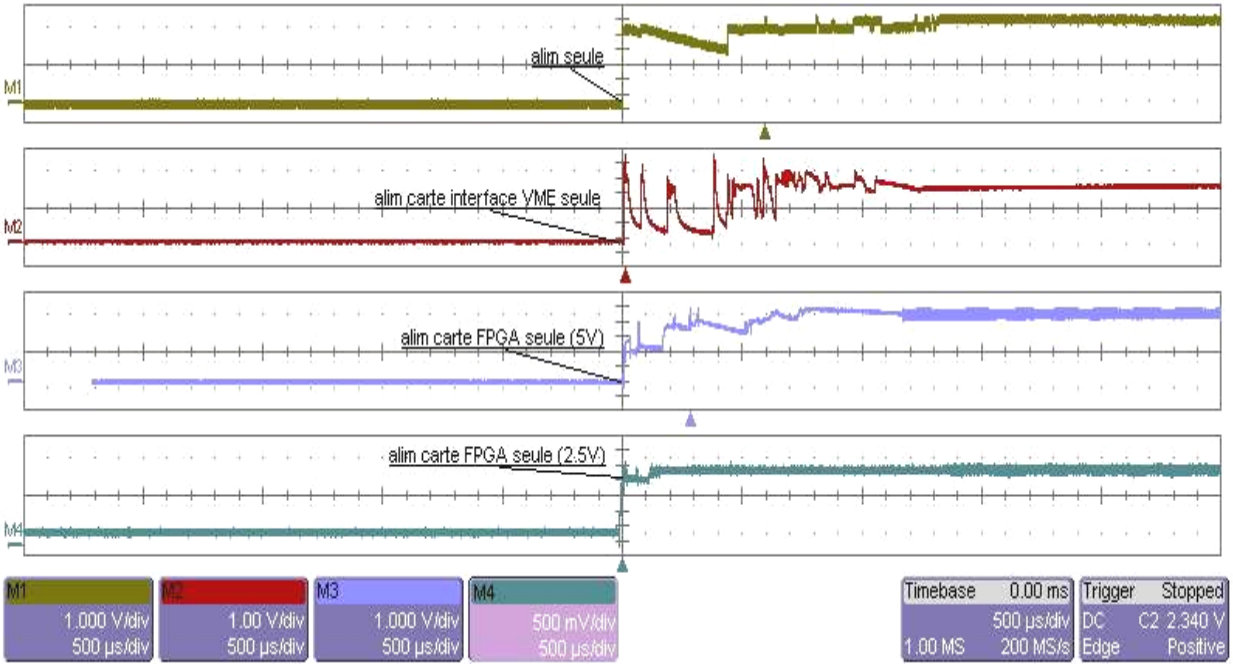


Figure 27 : Chronogramme des alimentations avant le filtrage du 5V.

Il a été également observé, plus précisément, des surtensions positives et négatives, pouvant endommager le FPGA. Un filtre à donc été placé sur la carte « interface VME », afin de supprimer ces surtensions et variations indésirables (voir « filtre alimentation 5V » page 25).

3.5 Le devenir du système

Suite aux différents tests réalisés sur le système complet, l'ensemble des cartes « time base board » et « interface VME » a été envoyé en Allemagne, à l'Université de Karlsruhe afin de poursuivre les tests sur le système de véto-muon.

La carte réalisée durant ce stage nécessitant de connaître son fonctionnement en vue de l'utiliser sur le système véto-muon, une documentation associée a été rédigée en anglais (disponible en annexe 7, page 87). Cette dernière reste succincte, et présente la carte essentiellement d'un point de vue fonctionnel.

Un retour d'expérience de la part de nos collaborateur en Allemagne, nous a permis de nous conforter dans la fiabilité de notre système ; les besoins de l'expérience sont satisfaits, et les options apportées à la carte « time base board » ont prouvé leurs utilités (signal « enable_latch », affichage en face avant des informations utiles…).

D'autre part, une saisie du schéma de câblage de la carte a été réalisée sous le logiciel « Board Design » de Cadence. Ceci permettra par la suite de réaliser une copie de la carte pour permettre d'éventuelles modifications et de disposer des mêmes outils de test, au sein de la collaboration EDELWEISS.

4 L'EFEBB

4.1 Qu'est-ce que l'EFEBB ?

L'EFEBB (Emetteur Fibre pour Emulation Boîte Bolo) est une carte de test permettant d'émuler le comportement de l'électronique « front end » (placée au plus proche des bolomètres).

L'électronique « front end » (correspondant aux boîtes bolo, en jaune, sur la figure 28, page 44), dont nous avons besoin pour mettre au point les nouvelles cartes de l'électronique d'acquisition (ICA, CC, IBB, voir la figure 28, page 44), n'est pas encore disponible ; de plus, son encombrement la rendrait difficile d'emploi pour la testabilité de notre système.

Pour mettre au point l'électronique d'acquisition, il suffit de disposer de fibres optiques délivrant des signaux simplifiés, avec un format identique à ceux de l'électronique « front end », d'où l'idée d'un émulateur compact pour délivrer de tels signaux.

L'ensemble de l'électronique d'acquisition à tester (figure 28, page suivante) est prévue pour accepter de traiter les données de 21 bolomètres dans un premier temps, puis 120 dans un deuxième temps. Nous ne détaillerons pas le fonctionnement de ces cartes.

Dans un tel émulateur, les cartes BB (Boîtes Bolo, en jaune) seront substituées par des cartes EFEBBs. Celles-ci devront émettre des données dédiées aux cartes IBB (Interface Boîte Bolo, en rouge) sur fibres optiques, à 6Mbits/s. Ces données seront retransmises sur les cartes CC (Contrôleur Châssis, en bleu), puis ICA (Interface Calculateur d'Acquisition) pour être traitées par des ordinateurs d'acquisition.

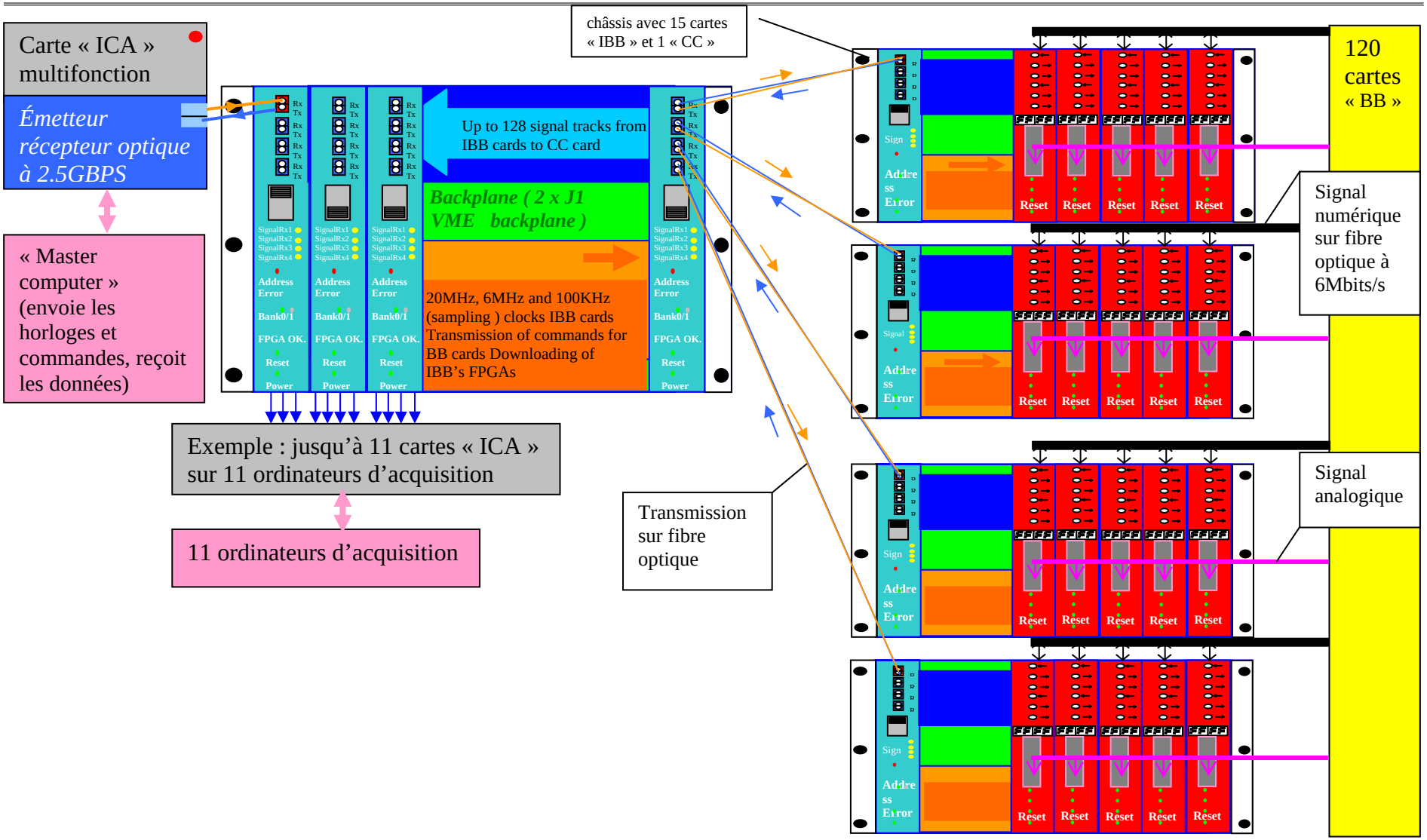


Figure 28 : Electronique d'acquisition numérique, en vue de l'étape 120 bolomètres.

4.2 Le principe

Afin de satisfaire aux différentes étapes évoluant vers une simulation du comportement de 120 bolomètres, et dans le but de réduire les coûts de développement, nous avons choisi de réaliser plusieurs cartes EFEBB au format européen (172x129mm), chacune acceptant six émetteurs fibre optique, correspondant à l'envoi de données de six bolomètres.

Les données envoyées sur ces fibres ne font que transiter par la carte EFEBB : dans un premier temps, une paire de connecteurs de type P160 (voir figure 26, page 40) permet, via le kit de développement Memec Design, d'utiliser le FPGA Xilinx VIRTEX II PRO. Celui-ci envoie sur l'EFEBB les six signaux correspondant aux données émises de six bolomètres, grâce au code VHDL décrit ci-après.

Dans un deuxième temps, plusieurs cartes EFEBB pourront être branchées sur un fond de panier (VME) où transiterons les données à envoyer. Ces données seront fournies par une carte CC (dont le fonctionnement a volontairement été modifié) disposant d'un FPGA Xilinx VIRTEX II PRO (2VP2-FF-672), connecté en émission ou en réception vers 128 lignes du fond de panier. Le FPGA permettra ainsi d'émuler le fonctionnement de 120 bolomètres, en envoyant sur le bus les 120 signaux, où les cartes EFEBB feront la réémission sur 120 fibres optiques.

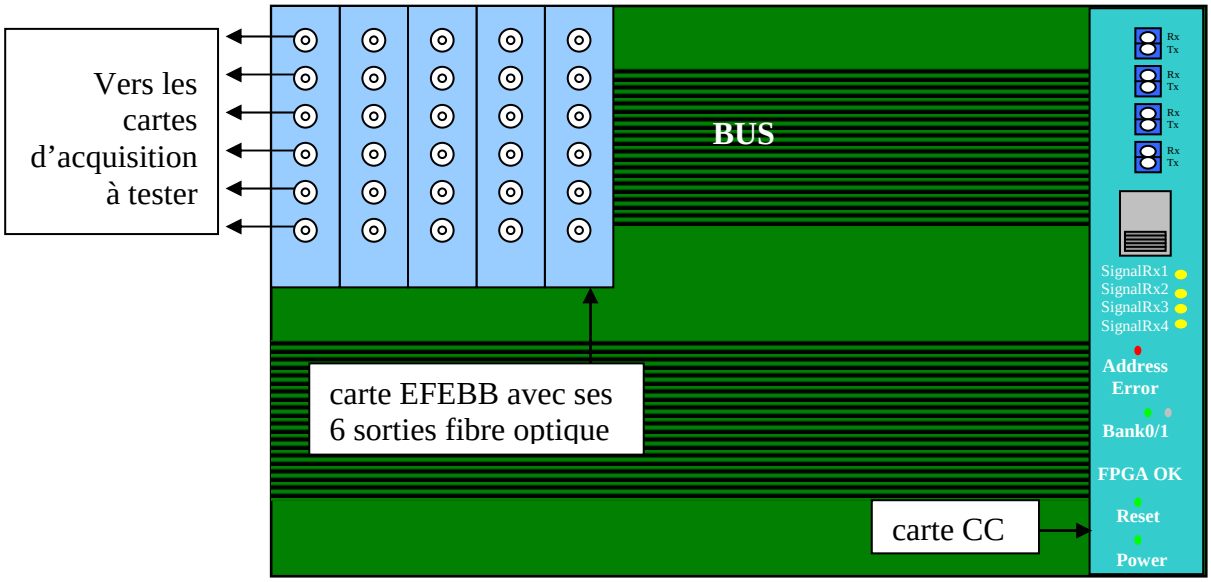


Figure 29 : Architecture utilisée pour l'étape de simulation du comportement de 120 bolomètres.

Les données que l'EFEBB doit renvoyer sont de deux types différents. Une commande (bouton poussoir sur le kit de développement, par exemple) permet d'effectuer le choix entre un mode d'acquisition et un mode d'identification.

Le premier mode consiste à ce que l'EFEBB simule l'envoi, vers les cartes d'acquisition IBB, d'une trame comportant les valeurs numérisées en provenance des bolomètres. Ne disposant pas de celles-ci, nous proposons d'envoyer les mots de même format, représentés sur la figure suivante :

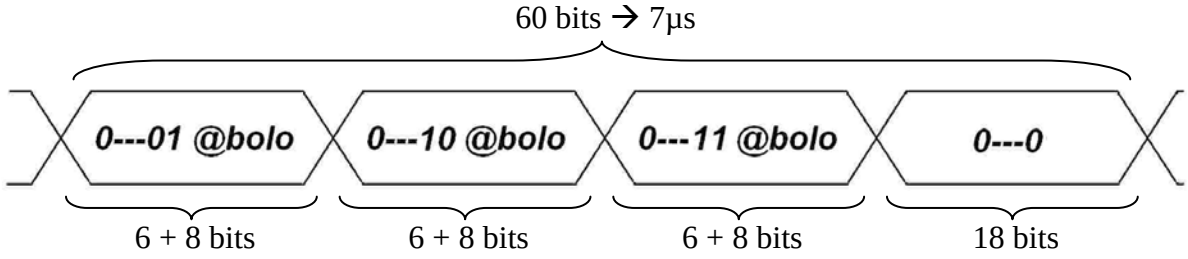


Figure 30 : Trame représentant la réponse d'un bolomètre en mode acquisition.

Le mode identification permet, en fonctionnement normal, de spécifier au système d'acquisition l'adresse du bolomètre sélectionné et le statut dans le lequel il se trouve. Pour notre système de test, nous figeons le statut à « 111 », ce qui nous donne une trame dont la forme est décrite à la figure suivante :

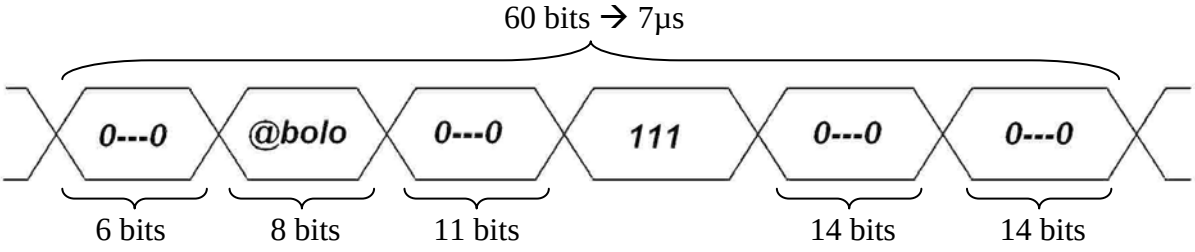


Figure 31 : Trame représentant la réponse d'un bolomètre en mode identification.

La mise au point de l'électronique d'acquisition nécessite de bien identifier les données après plusieurs opérations de sérialisation et dé-sérialisation. L'adresse contenue dans le mot est donc plus importante, pour cette opération de test, qu'une donnée physique.

Pour respecter le cas réel, les 60 bits constituant chacune des trames sont envoyés à 6Mbits/s, ce qui signifie que celles-ci sont retransmises toutes les 7µs.

Du point de vue fonctionnel, il doit être possible de sélectionner un bolomètre parmi les autres afin de lui imposer le mode voulu (acquisition ou identification). Les bolomètres non sélectionnés doivent continuer à émettre leurs données correspondantes au mode dans lequel ils se situaient.

La sélection d'un bolomètre se fait grâce à son adresse. Chaque bolomètre possède une adresse pouvant aller de 1 à 120. Afin de simuler un éventuel conflit d'adressage, notre système de test permet d'allouer la même adresse à plusieurs bolomètres. La sélection d'un bolomètre se fait grâce à un DIP switch (constitué de huit interrupteurs) placé sur le kit de développement. Dans le cas où nous souhaitons appliquer un mode identique à tous les bolomètres, l'adresse 0xFF doit être sélectionnée.

4.3 Architecture employée

L'envoi des trames « acquisition » et « identification » correspondant à chaque bolomètre, se fait en série. Nous pourrions pour cela utiliser les registres des CLBs disponibles dans les FPGA Xilinx ; seules les adresses des bolomètres différent et le nombre de registres utilisés pourrait ainsi être limité.

Cependant, les composants programmables que nous utilisons disposent également de RAM distribuée (répartie dans les CLBs) : 44Kbits pour le Xilinx XC2VP2 de la carte CC et 94Kbits pour le XC2VP4 du kit de développement.

L'utilisation de cette RAM se fait par simple déclaration. Par exemple, le signal RAM défini ci-dessous représente une RAM répartie, de 32 mots de 4 bits.

type ram_type is array (31 downto 0) of std_logic_vector (3 downto 0) ;
signal RAM : ram_type ;

Or, ce type de déclaration fait toujours appel aux CLBs, alors qu'il existe des blocs dédiés, appelés blocs RAM : 216 kbits pour le Xilinx XC2VP2 et 504Kbits pour le XC2VP4. Nous utilisons donc ceux-ci afin de mémoriser les données que doit envoyer chaque bolomètre.

Au lieu de réaliser un décalage de registres pour envoyer les données en série, nous réalisons le décalage de l'adresse de la RAM préalablement initialisée avec les trames à envoyées. Les dimensions de la RAM sont donc choisies en fonction du nombre de bolomètres et en fonction de la longueur de la trame à émettre (voir figure 32, page 48).

Pour notre première étape de test, nous souhaitons émettre les données émulant le traitement de six bolomètres. La RAM déclarée devrait avoir une largeur de 6 bits (pour 6 bolomètres) et une profondeur de 60 bits (correspondant à la longueur d'une trame). Le bloc RAM disponible le plus proche en taille possède une largeur de 8 bits pour une profondeur de 512 bits, ce qui correspond à une RAM de 4096bits.

L'incrémentation de l'adresse de la RAM se fait automatiquement et périodiquement de 0 à 59, à la fréquence de 6MHz, ce qui permet de délivrer l'ensemble des 60 bits constituant la trame de chaque bolomètre, toutes les 7µs.

La sélection du bit « i » de la RAM permet de récupérer les données du bolomètre numéro « i + 1 ».

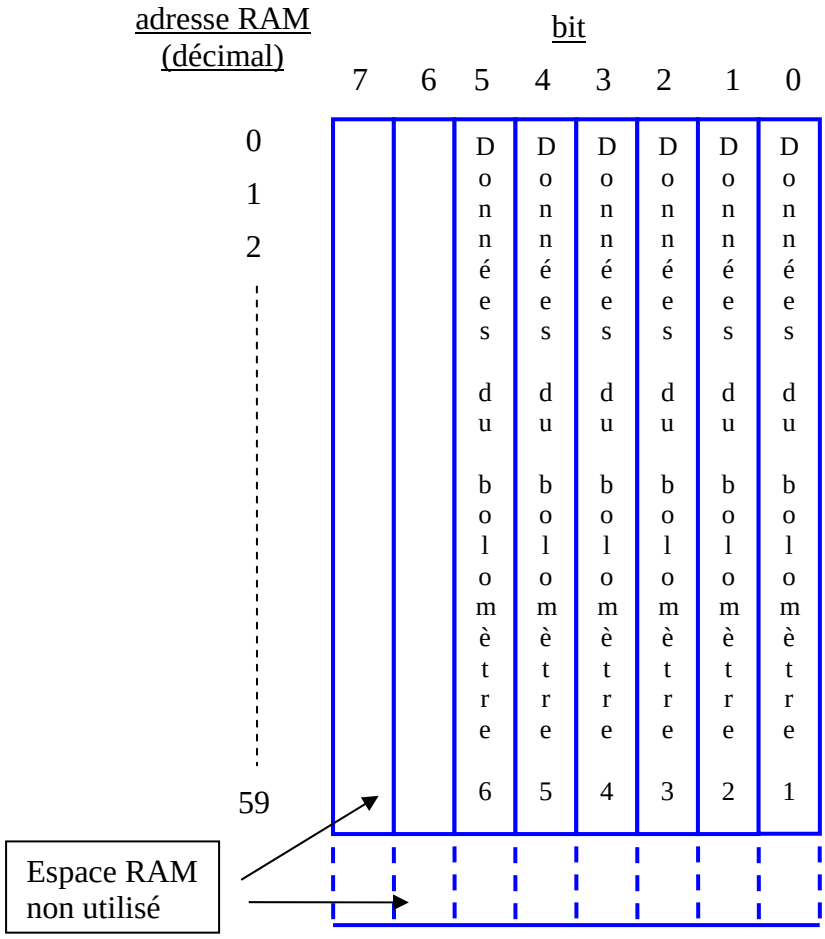


Figure 32 : Utilisation de la RAM pour l'envoi des données.

Etant donnée que les bolomètres peuvent répondre à deux modes différents (donc envoyer deux trames différentes), nous avons besoin de deux RAMs de 4096bits. Or, les FPGA Xilinx possèdent des blocs RAM doubles, dont la « RAMB4_S8_S8 » qui dispose de 4Kbits pour chacune des deux RAMs de 8 bits qui le composent. L'une de ces RAMs est donc dédiée à n'envoyer que les données correspondantes au mode « acquisition » alors que la seconde n'envoie que celles destinées au mode « identification ».

Un autre bloc VHDL permettra de choisir l'une ou l'autre RAM, en fonction de la commande du mode et en fonction de l'adresse du ou des bolomètres sélectionnés.

4.4 Description du code VHDL

La figure suivante décrit les différents blocs VHDL utilisés dans le but de fournir les données correspondantes à la réponse de six bolomètres (sortie notée « bolo(6:1) »).

Les entrées du système sont :

- L'horloge 125MHz présente sur le kit de développement. Celle-ci est appliquée en différentiel sur l'entrée du FPGA (minimisant ainsi les bruits tels que d'éventuels « jitters »).
- La commande, permettant de choisir entre le mode « identification » ou « acquisition ».
- Et l'adresse « sel_adr_bolo(8 :1) », pour un changement de mode du ou des bolomètres.

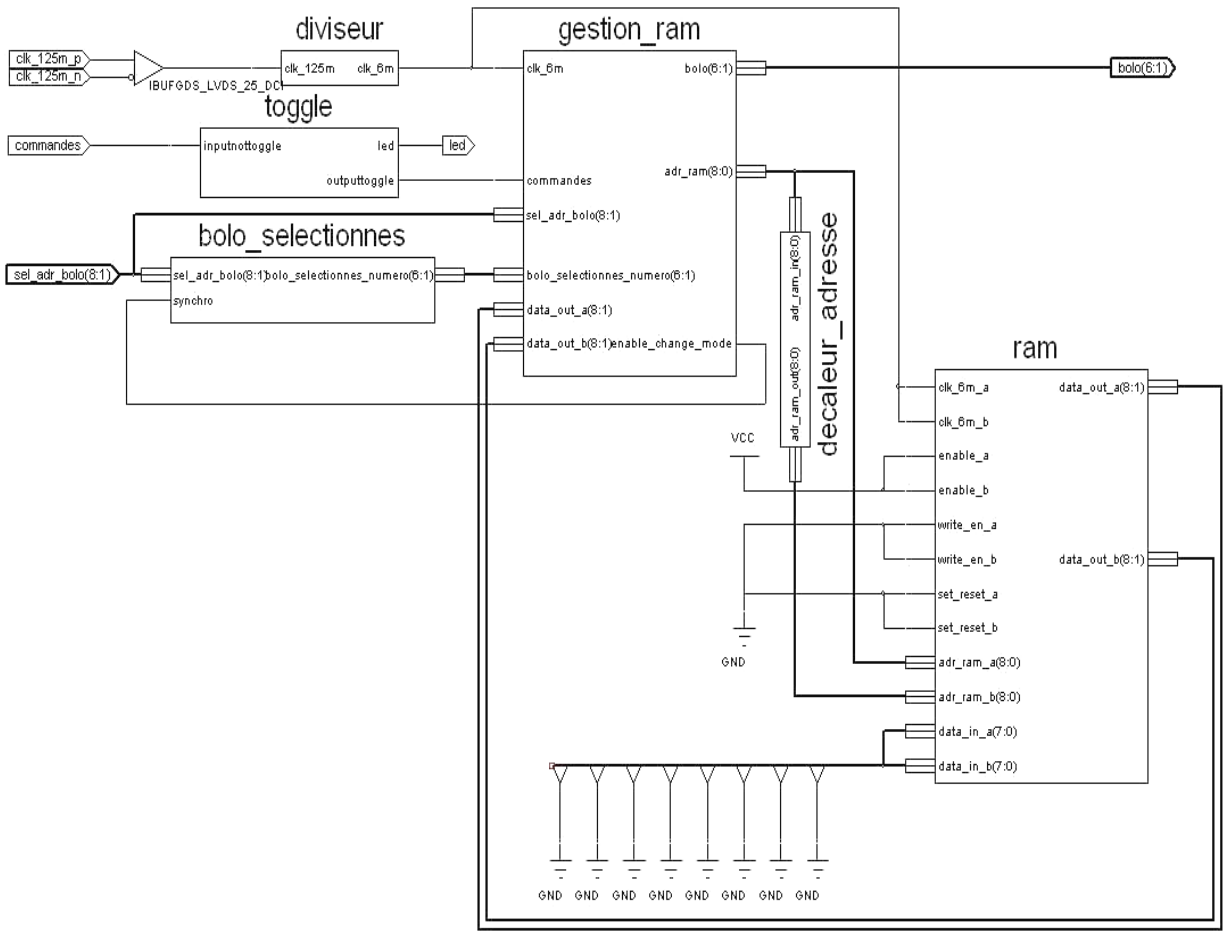


Figure 33 : Schéma-bloc du code VHDL.

- **Le bloc *diviseur* :**

Il permet de diviser la fréquence d'horloge 125MHz par 20 afin d'obtenir une fréquence proche de 6MHz (6.25Mhz), en maintenant un rapport cyclique de 50%. Ce bloc n'a pas été intégré à un autre afin de pouvoir facilement le modifier ou le supprimer, dans le cas où nous appliquerions une horloge différente à l'entrée du FPGA.

- **Le bloc *toggle* :**

Ce bloc n'est utile que dans le cas du test avec le kit de développement : la commande étant simulée par un bouton poussoir, le bloc *toggle* permet de maintenir l'état du bouton, tout en indiquant par un voyant lumineux le mode correspondant.

- **Le bloc *RAM* :**

La totalité du code VHDL de ce bloc est disponible en annexe 2 (page 71).

Ce bloc est une double RAM initialisée. Afin de l'utiliser en mode lecture uniquement, les entrées « data », « reset » et « write » sont maintenues à 0 ; et pour permettre son fonctionnement, l'entrée « enable » est maintenue à 1.

La sortie *data_out_a* délivre les trames des six bolomètres en mode acquisition, alors que la sortie *data_out_b* délivre celles en mode identification. Ces sorties recopient le contenu de la RAM aux endroits pointés par les adresses, qui sont envoyées sur les entrées respectives *adr_ram_a* et *adr_ram_b.*

Nous cherchons ici à instancier le bloc RAM « RAMB4_S8_S8 ».
Après avoir déclaré les entrées/sorties de notre RAM, il suffit de déclarer la composant à utiliser :

component RAMB4_S8_S8

La partie la plus importante est maintenant d'initialiser cette RAM ; les instructions INIT sont dédiées à cela : INIT_00 correspond aux 32 premiers mots de 8 bits de la RAM, INIT_01 aux 32 suivants…

Nous choisissons de déclarer la RAM correspondant au mode acquisition, sur les 256 premiers mots du bloc RAM (ce qui correspond à la moitié de sa taille) ; et la RAM correspondant au mode initialisation sur les 256 derniers.

Ainsi, seules les deux premières lignes d'initialisation de chaque RAM sont concernées (2x32 bits = 64 bits) étant donné la longueur d'une trame (60 bits). Soit l'initialisation suivante :

```
INIT_00 : bit_vector := X"000000003F3F0000000000" & "00111011" & "00100100" &
"00010100" & X"0000000000000000000000000000000000000000"; --RAMA 0 à 255
INIT_01:bit_vector:=X"00000000000000000003F0000000000" & "00111011" & "00100100"
& "00010100" & X"000000003F000000000000" & "00111011" & "00100100" &
"00010100"; --RAMA 256 à 511

INIT_08:bit_vector:=X"00000000000000000000000000000000000000000000000000000000
000000000"; --RAMB 2048 à 2303
 INIT_09 : bit_vector := X"00000000000000000000000000000000" & "00111011" &
"00100100" & "00010100" & X"00000000000000000000003F3F3F"; --RAMB 2304 à 2559
```

Les autres emplacements étant initialisés à 0.

Remarque :
Il est difficile de reconnaître la trame « acquisition » ou « identification » dans l'initialisation de la RAM correspondante, étant donné que les valeurs ont dues subir une transposition : ce que nous lisons verticalement dans la RAM (les trames à envoyer) doit être enregistré horizontalement dans celle-ci (voir figure 32, page 48).

Il ne reste plus qu'à instancier notre RAM à celle du modèle, pour compléter notre code :

```
U_RAMB4_S8_S8:  RAMB4_S8_S8
port map (
                CLKA    =>  clk_6M_A    ,
                CLKB    =>  clk_6M_B    ,
                ADDRA =>  adr_RAM_A ,
                …
        );
```

- **Le bloc *bolo_sélectionnés* :**

La totalité du code VHDL de ce bloc est disponible en annexe 3 (page 74).

Ce bloc sert à donner l'information du ou des bolomètres sélectionnés lorsqu'une adresse « sel_adr_bolo » est spécifiée. Le bloc « gestion_ram » a alors la possibilité d'affecter ses sorties « bolo » en fonction de l'adresse.

La sortie « bolo_selectionnes_numero » est un bus de 6 bits correspondant à la sélection des six bolomètres : si l'adresse « sel_adr_bolo » s'applique aux bolomètres 2 et 5, les sorties bolo_selectionnes_numero(2) et bolo_selectionnes_numero(5) passerons à 1.

Pour faire le lien entre les bolomètres et leurs adresses respectives, nous utilisons un tableau de six cases (pour six bolomètres) acceptant 8 bits chacune (adresse du bolomètre) :

```
type matrice is array(1 to 6) of std_logic_vector(8 downto 1);
signal adr_bolo_numero : matrice ;
```

Le bloc *bolo_selectionnes* doit connaître l'affectation des adresses, ce qui revient à initialiser le tableau précédent :

```
-- affectation des adresses correspondantes à chaque bolo
adr_bolo_numero(1) <= "00000100";
adr_bolo_numero(2) <= "00000100";
adr_bolo_numero(3) <= "00000011";
adr_bolo_numero(4) <= "00000100";
adr_bolo_numero(5) <= "00000101";
adr_bolo_numero(6) <= "00000110";
```

Une entrée *synchro* permet au bloc *gestion_ram* de recevoir l'information du ou des bolomètres sélectionnés, à l'instant approprié. Le bus de sortie du bloc *bolo_selectionnes* est sensible à cette entrée. Les bits du bus concernés par l'adresse sélectionnée passent à 1 en fonction de cette *sychro* alors que les autres sont à 0 pour indiquer que les bolomètres correspondants ne sont pas concernés :

```
process (synchro)
begin

   for i in 1 to 6 loop
      If sel_adr_bolo_integer = conv_integer(adr_bolo_numero(i)) Then
         bolo_selectionnes_numero(i) <= '1' ;
      else bolo_selectionnes_numero(i) <= '0' ;
      end if;
   end loop;

end process;

sel_adr_bolo_integer <= conv_integer(sel_adr_bolo) ; --sel_adr_bolo_integer est un signal
                                                       de type integer
```

- **Le bloc *gestion_RAM* :**

La totalité du code VHDL de ce bloc est disponible en annexe 4 (page 75).

La fonction de ce bloc est d'envoyer les données séries représentatives du comportement des six bolomètres en mode acquisition ou identification, en fonction des bolomètres et de la commande sélectionnés. Pour cela, ce bloc fait le choix de transmettre les données présentes en RAM « acquisition » ou en RAM « identification ».

Par défaut, à la mise sous tension, les bolomètres sont placés en mode acquisition. Si l'adresse 4 est sélectionnée, le bloc *bolo_selectionnes* indiquera que les bolomètres 1, 2 et 4 sont concernés. La sortie *bolo* du bloc *gestion_ram* réagit pour les bits 1, 2 et 4, en fonction de la commande désirée, alors que les autres bits gardent leur mode antérieur (voir figure suivante).

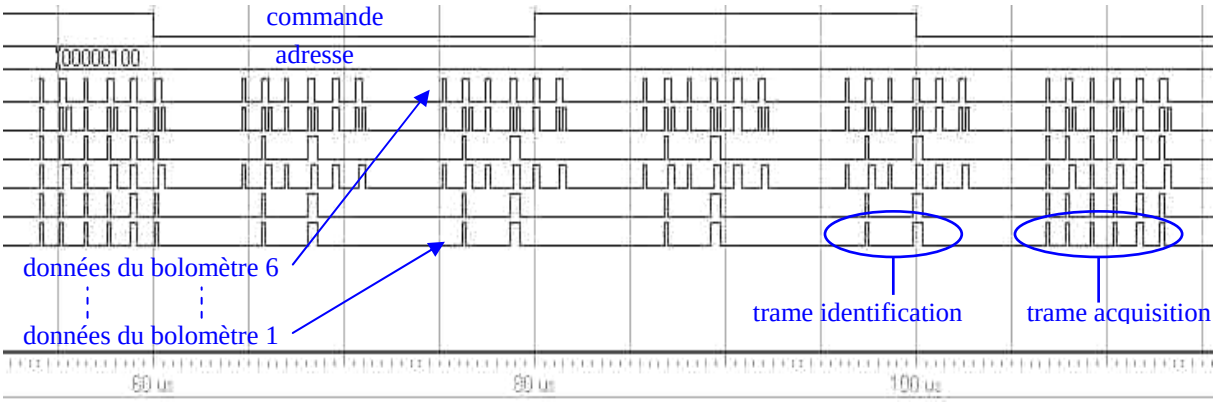


Figure 34 : Chronogramme des données émulées de six bolomètres.

Dans notre exemple, l'adresse sélectionnée est 4. Lorsque la commande passe à 0 (au temps 60µs), le mode passe en identification. Les bolomètres concernés (1, 2 et 4) envoient leurs donnés en répondant au mode identification, alors que les autres maintiennent leur mode acquisition. Un nouveau front descendant (au temps 100µs) de la commande permet au mode de passer en acquisition. Comme l'adresse est toujours positionnée en 4, les bolomètres 1, 2 et 4 reviennent en mode acquisition.

Si l'adresse est positionnée à 0xFF, cela est interprété comme étant une commande dédiée à tous les bolomètres. Le mode choisi s'applique donc aux six bolomètres dans notre cas.

Le bloc *gestion_ram* permet également au bloc *RAM* d'incrémenter les deux bus d'adresse à la fréquence de 6MHz, afin de délivrer les données à 6Mbits/s. Pour ne pointer que sur les zones utiles de la RAM, ces adresses sont incrémentées de 0 à 59 pour la première zone mémoire, et de même pour la seconde zone mémoire mais avec un décalage de 0x100. Ce décalage est réalisé par le bloc *decaleur_adresse*. L'incrémentation des adresses RAM se fait sur le front montant du 6MHz. Le changement de mode, quant à lui, se fait sur front descendant.

Si l'adresse sélectionnée se situe entre 1 et 6, le bloc *gestion_ram* scrute quels sont les bolomètres sélectionnés (entrée *bolo_selectionnes_numero*). Si la commande est à 0 (ce qui correspond au mode acquisition), la sortie *bolo* correspondante prend la valeur de la « RAM acquisition » :

```
if bolo_selectionnes_numero(i) = '1' then
        if commandes_latchee = '0' then
            bolo(i) <= DATA_OUT_A(i) ;
```

Puis le mode du bolomètre concerné est sauvegardé grâce au signal *mode_bolo_numero* :

```
if mode_bolo_numero(i) = '1' then
        mode_bolo_numero(i) <= '0' ;
end if ;
```

Dans le cas où la commande est à 1, la même logique est utilisée avec la « RAM identification ».

Il faut ensuite, pour les bolomètres non sélectionnés, maintenir le mode dans lequel ils se trouvaient, et continuer à envoyer les données. On se sert toujours du signal *mode_bolo_numero* où chaque mode de chaque bolomètre est sauvegardé :

```
----- pour les autres adresses, on maintient le mode précédent ------------
if mode_bolo_numero(i) = '0' then
        bolo(i) <= DATA_OUT_A(i);
else
        bolo(i) <= DATA_OUT_B(i);
end if;
```

Dans le cas où nous souhaitons commander tous les bolomètres dans un même mode, nous choisissons l'adresse 0xFF. Là encore, nous affectons à la sortie *bolo* les données de la « RAM acquisition » ou « RAM identification », selon la commande souhaitée :

```
------------ sinon même mode pour tous les bolo -----------------------------
elsif sel_adr_bolo_latchee = "11111111" then
        if commandes_latchee = '0' then
                bolo <= DATA_OUT_A(6 downto 1) ;
                mode_bolo_numero <= "000000" ;
        else
                bolo <= DATA_OUT_B(6 downto 1);
                mode_bolo_numero <= "111111" ;
        end if;
end if;
```

Remarque :
Afin de ne pas pouvoir changer de mode alors que les trames n'ont pas fini d'être envoyées, la commande ainsi que l'adresse de sélection des bolomètres sont mémorisés dans des registres (signaux *commandes_latchee* et *sel_adr_bolo_latchee*), jusqu'à obtenir l'autorisation d'un changement de mode (représenté par le signal *enable_change_mode*).

4.5 Le générateur de RAM

Dans le but de rendre notre système plus souple et afin de pouvoir envisager tous les cas possibles, une interface appelée « générateur de RAM », a été réalisée durant les derniers mois de stage.

La fonction de celle-ci est de permettre à l'utilisateur de changer les adresses des bolomètres, pour simuler d'éventuels conflits d'adressage par exemple. Or, nous avons vu dans la section précédente, qu'il est aisé d'affecter une adresse à un numéro de bolomètre (bloc *bolo_sélectionnés*), mais beaucoup plus difficile de modifier ces adresses au sein des trames émises, étant donné la transposition de données à réaliser lors de l'initialisation de la RAM.

Cette interface qui se doit d'être conviviale, sert donc à initialiser la RAM et à modifier le contenu du bloc *bolo_sélectionnés.*

Le choix de Visual Basic a été fait, offrant un environnement de programmation aisé et rapide, et dont l'aspect traditionnel « Windows » est facile d'utilisation.

La fenêtre principale offre la possibilité de saisir, dans le premier champ de texte, le numéro du bolomètre pour lequel on souhaite changer l'adresse ; cette dernière devant être enregistrée dans le second champ approprié (voir figure ci-dessous).

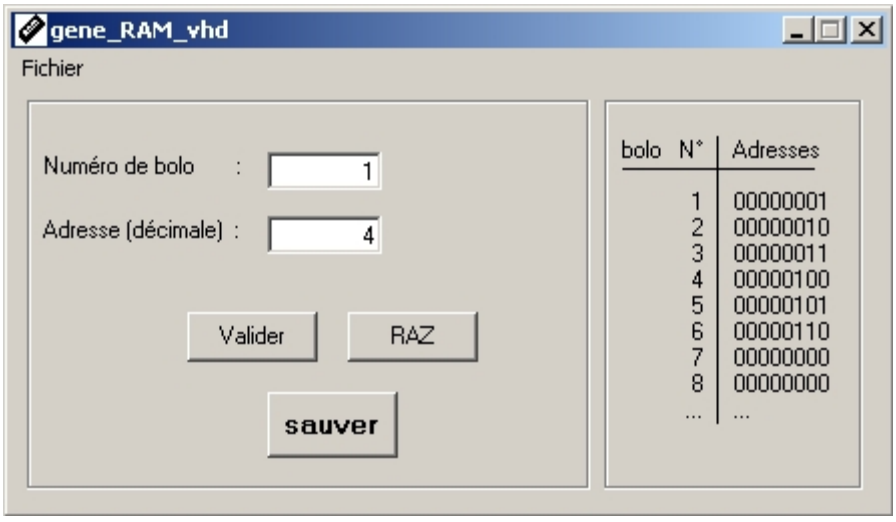


Figure 35 : "Générateur de RAM" réalisé sous Visual Basic.

L'utilisateur doit ensuite valider sa sélection par l'intermédiaire du bouton « valider », ce qui a pour effet de mettre à jour le tableau récapitulatif, situé dans le cadre de droite ; les champs de saisie sont automatiquement remis à zéro.

Il est également possible de réinitialiser toutes les adresses de tous les bolomètres, grâce au bouton « RAZ ». Ceci a pour effet d'affecter l'adresse 1 au bolomètre 1 et ainsi de suite jusqu'au bolomètre 6 (dans le cas de notre première étape de test).

La dernière étape consiste à enregistrer toute la saisie, par l'intermédiaire du bouton « sauver » ; ce qui génère le fichier RAM.vhd lié au bloc *RAM*, ainsi que le fichier bolo_selectionnes.vhd lié au bloc *bolo_selectionnes* précédemment décrits.

D'autre part, un menu « fichier » (voir figure suivante) permet d'accéder d'une part, au sous-menu « save as… » donnant accès à un explorateur, pour permettre l'enregistrement des deux fichiers VHDL à un emplacement spécifique ; et d'autre part au sous-menu « help », faisant apparaître une fenêtre d'aide concernant l'utilisation du programme.

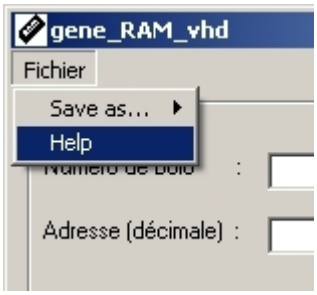


Figure 36 : Sous-menus disponibles.

L'ensemble du code de programmation est disponible en annexe 5 (page 78).

4.6 Etat d'avancement

4.6.1 Tâches réalisées

L'ensemble du code VHDL a été testé et validé avec les mêmes moyens de test que pour la carte « time base board » du véto-muon (simulateur ModelSim et kit de développement).

La seule difficulté majeure a été au niveau des tests effectués sur le kit de développement : certains signaux de sorties n'émettaient pas leurs trames correctement lors du passage d'un mode à l'autre. Pour le même code VHDL, la simulation sous ModelSim traduisait un comportement normal du système ; les bolomètres répondaient convenablement aux différentes commandes.

L'erreur se trouvait au niveau des lignes de code du bloc *gestion_ram*. Ces lignes, réalisant le changement de mode, se trouvaient dans le process de l'horloge 6Mhz, mais en dehors de la condition du front descendant. Or, le simulateur ModelSim réalise toute action sur front du process concerné, ce qui rendait l'erreur « transparente ».

Concernant la carte EFEBB, la saisie de schéma a été réalisée sous le logiciel « Board Design » de Cadence (disponible en annexe 6, page 85). Les émetteurs optiques choisis sont de type HFBR-1414T d'Agilent-Technologies.

Lors de l'utilisation de la carte CC, pour transmettre les données correspondantes à cent vingt bolomètres, la carte EFEBB doit permettre de choisir six signaux parmi les cent vingt, afin de les appliquer sur ses six fibres optiques. Cette sélection est rendue possible grâce à des « ponts CMS » entre le connecteur VME et les fibres optiques de la carte EFEBB (voir l'annexe 6, page 86).

4.6.2 Travail en cours et à venir

Le code VHDL étant validé, celui-ci ne subira que de légères modifications en fonction de l'évolution des besoins de test pour les cartes d'acquisition (CC, ICA, IBB…) : ajout d'un signal de synchronisme par rapport à l'électronique d'acquisition, insertion d'un compteur dans les trames émises…

D'autre part, le code existant n'étant conçu que pour un test sur six bolomètres, la programmation VHDL devra évoluer vers les étapes de 21 puis 120 signaux. Ceci n'aura d'influence que sur le stockage des données à émettre ; les blocs RAM disponibles dans les FPGAs n'étant pas extensibles à une longueur de mot de 120 bits, plusieurs solutions sont à envisager, comme par exemple la multiplication de blocs RAM.

Concernant la carte EFEBB, celle-ci est en cours de routage au sein du CEA, afin de pouvoir être conçue ensuite par un organisme extérieur. Cela permettra le lancement des premiers tests, en même temps que l'arrivée des nouvelles cartes d'acquisition : CC, IBB puis ICA. Les tests de l'ensemble du système d'acquisition sont prévus pour le mois de Décembre.

L'ensemble de l'électronique d'acquisition devrait être implanté dans le laboratoire sous terrain de Modane, dans le courant de l'été 2004, pour mettre en place la première étape de 21 bolomètres de l'expérience EDELWEISS II.

Conclusion

Après quelques difficultés rencontrées en début de stage lors de la conception de la carte « time base board », les collaborateurs allemands auxquels celle-ci était dédiée, ont exprimé toute satisfaction quant à son usage, au sein du système véto-muon.

La carte EFEBB est, quant à elle, en cours de réalisation. La fonction de celle-ci réalisée également autour d'un FPGA, répond aux besoins de test de la nouvelle électronique d'acquisition. Cependant, cette carte subit actuellement quelques modifications en fonction des nouveaux besoins de tests à prévoir.

L'ensemble de ces activités réalisées au sein du CEA m'ont permis de consolider mon expérience dans le développement de systèmes électroniques, centrés autour de l'environnement Xilinx et autour des FPGAs : aussi bien du point de vue du câblage de cartes et de composants, que du point de vue de la programmation, simulation et tests sur FPGA.

L'environnement de travail a également contribué au bon déroulement de ce projet : d'une part sur le plan technique, grâce aux moyens de développement mis à notre disposition, mais aussi sur le plan humain, avec une disponibilité des techniciens et ingénieurs, très appréciable.

D'autre part, le milieu de la recherche nécessitant une constante évolution du cahier des charges, les fonctions définitives des cartes à réaliser n'ont eu de cesse d'évoluer au fur et à mesure du temps et des différentes réunions de collaboration ; cela m'a permis de développer des facultés d'adaptation, ainsi que de compréhension des domaines d'activités aussi divers et variés que ceux de la cryogénie, physique, mécanique, informatique, optique …

Ces domaines, différents de celui de l'électronique, ont contribué à élargir ma vision sur un projet aussi vaste que celui proposé par l'expérience EDELWEISS ; rendant ainsi cette période de stage des plus enrichissantes.

Table des figures

Annexes

Annexe 1 :
VHDL de la carte « time base board »

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity cpt_pour_VME is
   Port (
                        sel_cpt          :      in  STD_LOGIC       ;
                        cpt_FO           :      in  STD_LOGIC       ;
                        reset_cpt        :      in  STD_LOGIC  ;
                        clk_125M           :  in  STD_LOGIC       ;
                        latch_ECL_n_ou_p  : in std_logic    ;
                        latch_face1, latch_face2, latch_face3, latch_face4, latch_face5,
latch_face6 : in std_logic ;
                        invert_latch_ECL  : in std_logic       ;
                        clk_16M_VME   :      in std_logic    ;
                        not_write, sel_0  :      in std_logic    ;
                        force_enable_latch:    in std_logic    ;
                        adr_signals       :      in std_logic_vector(3 downto 1)  ;
                        latch_LT1016   :       out std_logic   ;
                        test_enable_latch :    out std_logic   ;
                        test_serial_in    :    out std_logic   ;
                        led_cpt_FO        :    out std_logic   ;
                        led_latch_enable  :    out std_logic   ;
                        buzzer            :    out std_logic   ;
                        led_face1, led_face2, led_face3, led_face4, led_face5, led_face6 : inout
std_logic ;
                        serial_out        :    inout STD_LOGIC;
                        data              :    inout std_logic_vector(15 downto 0)

                );
end cpt_pour_VME;
```

```
architecture cpt_pour_VME_arch of cpt_pour_VME is

-------------------------------- signaux pour l'emission du compteur --------------------------------
constant pas_de_donnee: STD_LOGIC_VECTOR (49 downto 0) :=
"11111111111111111111111111111111111111111111111111"   ;
constant un : STD_LOGIC_VECTOR (47 downto 0) :=
"000000000000000000000000000000000000000000000001"                                   ;
constant zero : STD_LOGIC_VECTOR (47 downto 0) :=
"000000000000000000000000000000000000000000000000"                          ;

signal trame : STD_LOGIC_VECTOR ( 49 downto 0 )      ;
signal trame_I : STD_LOGIC_VECTOR ( 49 downto 0 )   ;
signal enable_shift : STD_LOGIC                                ;
-----------------------------------------------------------------------------------------------------

-------------------------------- signaux pour la reception du compteur --------------------------------
signal i : INTEGER := 0       ;         --
signal j : INTEGER := 0       ;         ----
signal k : INTEGER := 0       ;         ------  compteurs
signal l : INTEGER := 0       ;         ----
signal m : INTEGER := 0      ;         --
signal wait_start : STD_LOGIC := '0'                  ;
signal serial_in  : std_logic                                 ;
signal enable_decode : STD_LOGIC := '0'             ;
signal latch : STD_LOGIC := '0'                          ;
signal latch_ECL : STD_LOGIC := '0'                  ;
signal upgrade_cpt_latche : STD_LOGIC := '0'      ;
signal cpt_a_16MHz : INTEGER := 0                     ;
signal cpt_bit_a_1 : INTEGER := 0            ;         -- pour savoir si l'on se situe avant le bit
de start
signal cpt_des_64bits : INTEGER := 0        ;         -- pour compter les 64 bits de la trame
signal trame_decodee : STD_LOGIC_VECTOR (47 downto 0) := zero ;
signal cpt_latche : std_logic_vector(47 downto 0):= zero      ;
-----------------------------------------------------------------------------------------------------

-------------------------------- signaux pour le VME ------------------------------------------------
signal enable_latch : STD_LOGIC := '0'                ;
signal reset_enable_latch : STD_LOGIC := '0'        ;
signal or_enable_latch : STD_LOGIC := '0'            ;
-----------------------------------------------------------------------------------------------------

-------------------------------- autres signaux -----------------------------------------------------
signal tempo_buzzer  : INTEGER := 0        ;
signal enable_sound : STD_LOGIC := '0'    ;
signal clk_200Hz       : STD_LOGIC := '0'  ;
signal led_on            : STD_LOGIC := '0'  ;
signal tempo_led       : INTEGER := 0        ;
-----------------------------------------------------------------------------------------------------
```

```
begin

        process( tempo_buzzer, clk_125M, serial_in, sel_cpt, latch, latch_ECL,
invert_latch_ECL, wait_start )
                begin
                        if rising_edge(clk_125M) then -- attention, ce signal est divisé par 2 en
fréquence

--**************************************************************************--
--******************** pour l'émission du compteur  ****************************--
--**************************************************************************--

                                -------- generation du signal 100KHz -------------
                                i <= i + 1 ;
                                if i = 624 then
                                        i <= 0 ;
                                end if;
                                -------- generation du signal 16MHz-------------
                                j <= j + 1 ;
                                if j = 3 then
                                        j <= 0 ;
                                end if;
                                -------- generation du signal 200Hz -------------
                                if i = 0 then
                                        l <= l + 1 ;
                                        if l = 499 then
                                                l <= 0 ;
                                        end if;
                                end if;
                                -------- generation de l'horloge 200Hz -------------
                                if l = 0 then
                                        clk_200Hz <= '0';
                                elsif l = 249 then
                                        clk_200Hz <= '1';
                                end if;


                                ---- incrementation du compteur a la frequence 100KHz ----
                                if reset_cpt = '1' then
                                        trame <= '1' & zero & '0';      -- bit de stop & valeur
copteur & bit de start
                                elsif i = 312 then                              -- si front montant
de 100KHz
                                        trame <= '1' & (trame(48 downto 1) + un) & '0' ;
                                else
                                        trame <= trame ;
                                end if;
```

```
                        ---------- affectation de la trame ----------
                        if reset_cpt = '1'      then
                                trame_I <= pas_de_donnee ;
                        elsif j = 2 then                        -- sur chaque front
du 16MHz
                                if enable_shift = '0' then
                                        trame_I <= trame;       -- on load
                                else
                                        trame_I <= ('1' & trame_I(49 downto 1)); -- on
shift
                                end if;
                        end if;
--*************************************************************************--
--*************************************************************************--



--*************************************************************************--
--******************** pour la réception du compteur  ***********************--
--*************************************************************************--
                        ---------------------------------------------------------
                        ------- on compte le nombre de bits consecutifs à 1  ----
                        ------- afin de trouver le début (bit start) de la trame ---
                        ---------------------------------------------------------
                        if serial_in = '1' and enable_decode = '0' then
                                -- generation du 16MHz synchronisé sur les bits à 1 de la
trame --
                                if cpt_a_16MHz >= 3 then
                                        cpt_a_16MHz <= 0;
                                        cpt_bit_a_1 <= cpt_bit_a_1 + 1;
                                else    cpt_a_16MHz <= cpt_a_16MHz + 1;
                                end if;
                                -------- on compte jusqu'a 49 bits consecutifs a 1 --------
                                if cpt_bit_a_1 = 49 then
                                        cpt_bit_a_1 <= 0;
                                end if;
                                -- si 49 bits consecutifs a 1 alors on attend le bit de start
                                if cpt_a_16MHz = 3 and cpt_bit_a_1 = 48 then
                                        wait_start <= '1' ;
                                end if;
                        ------- si les bits sont a 0, on ne compte pas ------
                        else    cpt_bit_a_1 <= 0;
                                cpt_a_16MHz <= 0;
                                ------------------------------------------------
                                ---------- decodage de la trame ----------
                                ------------------------------------------------
                                if enable_decode = '1' then
```

```
                                        --- generation de 16MHz synchronisé sur le bit start ---
                                                if cpt_a_16MHz >= 3 then
                                                        cpt_a_16MHz <= 0;
                                                else    cpt_a_16MHz <= cpt_a_16MHz + 1;
                                                end if;


                                                ------------ capture du bit de la trame --------------
                                                if cpt_a_16MHz = 1 then
                                                                cpt_des_64bits <= cpt_des_64bits
+ 1;
                                                                trame_decodee <= serial_in &
trame_decodee(47 downto 1);
                                                end if;

                                                ------ si l'on a decodé les 50 bits, on arrete ------
                                                if cpt_des_64bits = 49 then
                                                        enable_decode <= '0';
                                                        cpt_des_64bits <= 0;
                                                end if;
                                        else cpt_a_16MHz <= 0 ;
                                        end if;
                                end if;


                                ---------- mise à jours du compteur latché ----------
                                if enable_decode = '0' and upgrade_cpt_latche = '1' then
                                        upgrade_cpt_latche <= '0' ;
                                        cpt_latche <= trame_decodee ;
                                end if;

                        end if;


                        ---------- permission de decoder la trame ----------
                        ---------- (sensible uniquement à serial_in) ----------
                        if serial_in = '0' and wait_start = '1' then
                                wait_start <= '0'       ;
                                enable_decode <= '1';
                                cpt_a_16MHz <= 0    ;
                        end if;

                        ---------- mémorisation du latch  ----------
                        ---------- (sensible à "latch" uniquement) ----------
                        if latch = '1' then
                                upgrade_cpt_latche <= '1';
                        end if;
--**************************************************************************--
--**************************************************************************--
```

```
------ demande de reset de l'enable_latch qui provient du VME -----------
if latch = '1' and latch_ECL = '1' then
        reset_enable_latch <= '1';
elsif latch_ECL = '0' then
        reset_enable_latch <= '0';
end if;
-------------------------------------------------------------------------------


-- selection du compteur fibre optique ou du loopback avec l'interrupteur --
        case sel_cpt is
            when '1' =>
                    serial_in <= serial_out ;
                    led_cpt_FO <= '0';              -- led active sur état bas
                    latch_LT1016 <= '1';    -- on empeche la réception de la
FO sur le comparateur
            when '0' =>
                serial_in <= cpt_FO ;
                    latch_LT1016 <= '0';  -- on permet la réception de la FO
sur le comparateur
                    if (m > 0 and m < 99) then
                        led_cpt_FO <= '0' ;
                    else led_cpt_FO <= '1' ;
                    end if ;
                when others =>
                    null;
        end case;
        -------------------------------------------------------------------------------

        ----------------- commande d'inversion du latch_ECL ----------------------
        if invert_latch_ECL = '1' then
                latch_ECL <= not latch_ECL_n_ou_p ;
        else latch_ECL <= latch_ECL_n_ou_p ;
        end if;
        -------------------------------------------------------------------------------


        ----------------- commande du buzzer lors d'un latch_ECL ----------------
        if latch_ECL = '1' then
                enable_sound <= '1' ;    -- envoie du son sur le buzzer
        end if ;

        if enable_sound = '1' then
                if (l > 0 and l < 124) or (l > 250 and l < 375) then -- fréquence
du son
                    buzzer <= '1' ;    -- avec rapport cyclique modifiable
```

```
                else
                        buzzer <= '0' ;
                end if ;
        elsif enable_sound = '0' then
                buzzer <= '0' ;
        end if ;

        if tempo_buzzer = 99 then       -- durée du son
                enable_sound <= '0' ;
                buzzer <= '0'           ;
        end if ;
        ------------------------------------------------------------------------------
end process;



process (clk_200Hz)
begin
        if rising_edge(clk_200Hz) then
                ---------- temps d'emission sur le buzzer ----------
                if enable_sound = '1' then
                        tempo_buzzer <= tempo_buzzer + 1 ;
                elsif enable_sound = '0' then
                        tempo_buzzer <= 0 ;
                end if ;

                ---------- generation du signal 1Hz ----------
                m <= m + 1 ;
                if m = 199 then
                        m <= 0 ;
                        tempo_led <= tempo_led + 1 ;
                end if ;

                ---------- tempo pour l'extinction des LEDs ---------
                if tempo_led = 5 then
                        tempo_led     <= 0 ;
                end if;

        end if ;
end process;
```

```
--*************************************************************************--
--******************** pour la partie "gestion VME" ***************************--
--*************************************************************************--
        process (clk_16M_VME, latch_face1, latch_face2, latch_face3, latch_face4,
latch_face5, latch_face6)
            begin

                if rising_edge(clk_16M_VME) then

                    ---------- le VME lit les valeurs du compteur latché ----------
                    if not_write = '1' and sel_0 = '0' then  -- le bus VME est en
lecture

                        case adr_signals is
                            when "000" =>
                                data <=  "0000000000000000" ;
    --en vue d'un compteur 64 bits
                            when "001" =>
                                data <= cpt_latche(47 downto 32);
                            when "010" =>
                                data <= cpt_latche(31 downto 16);
                            when "011" =>
                                data <= cpt_latche(15 downto 0);
                            when others =>
                                null ;
                        end case;
                    end if;

                    --------------- réception d'un enable_latch provenant du VME ----
                    if not_write = '0' and sel_0 = '0' and adr_signals(3) = '1' then --
le bus VME est en écriture
                        data <= "ZZZZZZZZZZZZZZZZ" ; -- on attend une data
                        if force_enable_latch = '0' then
                            led_face1 <= '1';  -- on etteinds les leds
                            led_face2 <= '1';
                            led_face3 <= '1';
                            led_face4 <= '1';
                            led_face5 <= '1';
                            led_face6 <= '1';
                        end if;
                        if data(0) = '1' then
                            enable_latch <= '1';
                        end if;
                    end if;

                    -- si l'enable_latch est rendu optionnel, on temporise l'extinction
des leds --
                    if led_on = '1' and force_enable_latch = '1' then
```

```
                                    if tempo_led = 5 then
                                        led_face1 <= '1';  -- on etteinds les leds
                                        led_face2 <= '1';
                                        led_face3 <= '1';
                                        led_face4 <= '1';
                                        led_face5 <= '1';
                                        led_face6 <= '1';
                                    end if;
                                end if;

                                -- reset de l'enable_latch en vue d'en attendre un nouveau, sinon
                                -- on prend en compte l'enable_latch du VME qd il vient
d'arriver, uniquement -----
                                if reset_enable_latch = '1' then
                                    enable_latch <= '0' ;
                                end if;
                            end if;

                            ---------------------- maintient des signaux de latch -------------------------
                            if latch_face1 = '1' then
                                led_face1 <= '0';       -- active sur l'état bas
                            end if;
                            if latch_face2 = '1' then
                                led_face2 <= '0';       -- active sur l'état bas
                            end if;
                            if latch_face3 = '1' then
                                led_face3 <= '0';       -- active sur l'état bas
                            end if;
                            if latch_face4 = '1' then
                                led_face4 <= '0';       -- active sur l'état bas
                            end if;
                            if latch_face5 = '1' then
                                led_face5 <= '0';       -- active sur l'état bas
                            end if;
                            if latch_face6 = '1' then
                                led_face6 <= '0';       -- active sur l'état bas
                            end if;
                            ------------------------------------------------------------------------------

                    end process;
--*****************************************************************************--
--*****************************************************************************--
```

```
        ------------------------- Concurrent Assignement ----------------------------------

        ------------------------- pour la partie émission du compteur ----------------------
        -- on ne shift pas pendant les premieres 64ns de chaque front montant du 100KHZ
        enable_shift <= '0' when (i >= 313 and i <=316) else '1'     ;
        serial_out <=  trame_I(0)                                                    ;
        ------------------------------------------------------------------------------------

        ------------------------- pour la partie reception du compteur ----------------------
        test_enable_latch <= or_enable_latch    ;
        test_serial_in <= serial_in                  ;
        ------------------------------------------------------------------------------------

        ---------------------------------- pour la partie VME --------------------------------
        latch <= latch_ECL and or_enable_latch     ;
        led_latch_enable <= not or_enable_latch ;
        or_enable_latch <= enable_latch or force_enable_latch ;   -- pour rendre optionnel le
signal enable_latch
        led_on <= not (led_face1 or led_face2 or led_face3 or led_face4 or led_face5 or
led_face6) ;
        ------------------------------------------------------------------------------------

end;
```

Annexe 2 :
VHDL de la carte du bloc *RAM*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;
library UNISIM;
use UNISIM.VCOMPONENTS.ALL;

entity RAM is
    Port (
          clk_6M_A       :  in  STD_LOGIC              ;
          clk_6M_B       :  in  STD_LOGIC              ;
          adr_RAM_A      :  in  STD_LOGIC_VECTOR (8 downto 0)   ;
          adr_RAM_B      :  in  STD_LOGIC_VECTOR (8 downto 0)   ;
          DATA_IN_A      :  in  STD_LOGIC_VECTOR (7 downto 0)   ;
          DATA_IN_B      :  in  STD_LOGIC_VECTOR (7 downto 0)   ;
          ENABLE_A       :  in  STD_LOGIC              ;
          ENABLE_B       :  in  STD_LOGIC              ;
          WRITE_EN_A     :  in  STD_LOGIC              ;
          WRITE_EN_B     :  in  STD_LOGIC              ;
          SET_RESET_A    :  in  STD_LOGIC              ;
          SET_RESET_B    :  in  STD_LOGIC              ;
          DATA_OUT_A     :  out STD_LOGIC_VECTOR (8 downto 1)  ;
          DATA_OUT_B     :  out STD_LOGIC_VECTOR (8 downto 1)
         );
end RAM ;



architecture RAM_arch of RAM is


----- Component RAMB4_S8_S8 -----
component RAMB4_S8_S8


  generic (
     -- mode acquisition
     INIT_00 : bit_vector := X"000000003F3F0000000000" & "00111011" & "00100100" &
"00010100" & X"0000000000000000000000000000000000000000"; --RAMA 0 à 255
     INIT_01 : bit_vector := X"00000000000000000003F0000000000" & "00111011" &
"00100100" & "00010100" & X"000000003F000000000000" & "00111011" & "00100100"
& "00010100"; --RAMA 256 à 511
```

```
    INIT_02 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"; --RAMA
512 à 767
    INIT_03 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"; --RAMA
768 à 1023
    INIT_04 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"; --RAMA
1024 à 1278
    INIT_05 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"; --RAMA
1279 à 1533
    INIT_06 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"; --RAMA
1534 à ...
    INIT_07 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"; --RAMA

    -- mode identification
    INIT_08 : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"; --RAMB
2048 à 2303
    INIT_09 : bit_vector := X"00000000000000000000000000000000" & "00111011" &
"00100100" & "00010100" & X"000000000000000000003F3F3F"; --RAMB 2304 à 2559
    INIT_0A : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"; --RAMB
2600 à 2815
    INIT_0B : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"; --RAMB
2816 à ...
    INIT_0C : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"; --RAMB
    INIT_0D : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"; --RAMB
    INIT_0E : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"; --RAMB
    INIT_0F : bit_vector :=
X"0000000000000000000000000000000000000000000000000000000000000000"  --RAMB
3840 à 4095
      );

port (
      DIA   : in STD_LOGIC_VECTOR (7 downto 0);
      DIB   : in STD_LOGIC_VECTOR (7 downto 0);
      ENA   : in STD_logic;
      ENB   : in STD_logic;
      WEA   : in STD_logic;
      WEB   : in STD_logic;
      RSTA  : in STD_logic;
```

```
      RSTB   : in STD_logic;
      CLKA   : in STD_logic;
      CLKB   : in STD_logic;
      ADDRA  : in STD_LOGIC_VECTOR (8 downto 0);
      ADDRB  : in STD_LOGIC_VECTOR (8 downto 0);
      DOA    : out STD_LOGIC_VECTOR (7 downto 0);
      DOB    : out STD_LOGIC_VECTOR (7 downto 0)
    );
end component;



begin

-- bloc ram instantiation
U_RAMB4_S8_S8:  RAMB4_S8_S8
port map (
      CLKA    =>  clk_6M_A    ,
      CLKB    =>  clk_6M_B    ,
      ADDRA   =>  adr_RAM_A ,
      ADDRB   =>  adr_RAM_B ,
      DIA     =>  DATA_IN_A ,
      DIB     =>  DATA_IN_B ,
      ENA     =>  ENABLE_A    ,
      ENB     =>  ENABLE_B    ,
      WEA     =>  WRITE_EN_A ,
      WEB     =>  WRITE_EN_B ,
      RSTA    =>  SET_RESET_A,
      RSTB    =>  SET_RESET_B,
      DOA     =>  DATA_OUT_A(8 downto 1) ,
      DOB     =>  DATA_OUT_B(8 downto 1)
      );

end;
```

Annexe 3 :

VHDL de la carte du bloc bolo_selectionnés

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity bolo_selectionnes is
   Port (
         synchro                    :  in  std_logic := '0'              ;
         sel_adr_bolo               :  in  STD_LOGIC_VECTOR (8 downto 1)   ;
         bolo_selectionnes_numero:  out STD_LOGIC_VECTOR (6 downto 1) := "000000"
        );
end bolo_selectionnes;


architecture bolo_selectionnes_arch of bolo_selectionnes is

type matrice is array(1 to 6) of std_logic_vector(8 downto 1);
signal adr_bolo_numero : matrice ;
signal sel_adr_bolo_integer : integer ;

begin

-- affectation des adresses correspondantes à chaques bolo
adr_bolo_numero(1) <= "00000100";
adr_bolo_numero(2) <= "00000100";
adr_bolo_numero(3) <= "00000011";
adr_bolo_numero(4) <= "00000100";
adr_bolo_numero(5) <= "00000101";
adr_bolo_numero(6) <= "00000110";

process (synchro)
begin

   for i in 1 to 6 loop
      If sel_adr_bolo_integer = conv_integer(adr_bolo_numero(i)) Then
         bolo_selectionnes_numero(i) <= '1' ;
      else bolo_selectionnes_numero(i) <= '0' ;
      end if;
   end loop;

end process;

sel_adr_bolo_integer <= conv_integer(sel_adr_bolo) ;

end;
```

Annexe 4 :
VHDL de la carte du bloc gestion_RAM

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity gestion_RAM is
	Port (
			clk_6M		:	in  STD_LOGIC			;
			commandes	:	in  STD_LOGIC := '0'		;
			DATA_OUT_A	:	in  STD_LOGIC_VECTOR (8 downto 1);
			DATA_OUT_B	:	in  STD_LOGIC_VECTOR (8 downto 1);
			sel_adr_bolo	:	in  STD_LOGIC_VECTOR (8 downto 1);
			bolo_selectionnes_numero : in STD_LOGIC_VECTOR (6 downto 1) ;
			adr_RAM		:	out STD_LOGIC_VECTOR (8 downto 0);
			enable_change_mode : inout std_logic := '0'		;
			bolo		:	out STD_LOGIC_VECTOR (6 downto 1)
:= "000000"
	);
end gestion_RAM;



architecture gestion_RAM_arch of gestion_RAM is

-------------------------------------------------------------------------------
signal adr_RAM_I : STD_LOGIC_VECTOR (8 downto 0) :=
CONV_STD_LOGIC_VECTOR (0, 9) ;
signal i, j				: integer := 1			;
signal sel_adr_bolo_latchee	: std_logic_vector (8 downto 1)	;
signal commandes_latchee	: std_logic := '0'		;
signal mode_bolo_numero		: std_logic_vector(6 downto 1)  :=
CONV_STD_LOGIC_VECTOR (0, 6)  ;
-------------------------------------------------------------------------------

begin

process (clk_6M)
	begin

		-------- met à jour le mode uniquement après la fin d'envoi d'une trame ----------
		if enable_change_mode = '1' then
			commandes_latchee <= commandes ;
		end if;
		-------------------------------------------------------------------------------
```

```
		if rising_edge(clk_6M) then

			------ shift de l'adresse de la RAM correspondant au shift des trames de chaque bolo ----------
			adr_RAM_I <= adr_RAM_I - 1 ;
			if adr_RAM_I = 0 then
				adr_RAM_I <= CONV_STD_LOGIC_VECTOR (59, 9) ;
			elsif adr_RAM_I = 59 then
				enable_change_mode <= '1' ;		--\ pour ne changer de mode que
				sel_adr_bolo_latchee <= sel_adr_bolo ; --/ lorsque les trames ont fini d'être envoyées
			else
				enable_change_mode <= '0' ;
			end if;
			-------------------------------------------------------------------------------

		end if;


		if falling_edge(clk_6M) then

			---------- changement de mode pour une adresse particulière --------------
			if ("00000001" <= sel_adr_bolo_latchee and sel_adr_bolo_latchee <= "00000110") then
				for i in 1 to 6 loop
					if bolo_selectionnes_numero(i) = '1' then
						if commandes_latchee = '0' then
							bolo(i) <= DATA_OUT_A(i);

							if mode_bolo_numero(i) = '1' then
								mode_bolo_numero(i) <= '0';
							end if;
						else
							bolo(i) <= DATA_OUT_B(i);
							if mode_bolo_numero(i) = '0' then
								mode_bolo_numero(i) <= '1';
							end if;
						end if;
					else
			----- pour les autres adresses, on maintient le mode précédent ------------
						if mode_bolo_numero(i) = '0' then
							bolo(i) <= DATA_OUT_A(i);
						else
							bolo(i) <= DATA_OUT_B(i);
						end if;
					end if;
				end loop;
```

```
                elsif sel_adr_bolo_latchee = "11111111" then
                ------------ sinon même mode pour tous les bolo ----------------------------
                        if commandes_latchee = '0' then
                                bolo <= DATA_OUT_A(6 downto 1) ;
                                mode_bolo_numero <= "000000" ;
                        else
                                bolo <= DATA_OUT_B(6 downto 1);
                                mode_bolo_numero <= "111111" ;
                        end if;

                end if;
        end if;

    end process;

adr_RAM <= adr_RAM_I ;

end;
```

Annexe 5 :
Code Visual Basic du « générateur de RAM »

```
Dim matrice(1 To 8, 1 To 8) As Integer
Dim adr_transposee(1 To 8) As String * 8
Dim bolo(1 To 8) As String * 8


Private Sub Command1_Click()       ‘---- bouton sauver ----

'------------------- crée le fichier VHDL modélisant la RAM ----------------------------
Open "RAM.vhd" For Output As #1   ' Ouvre le fichier en écriture.
Print #1, "library IEEE;"
Print #1, "use IEEE.STD_LOGIC_1164.ALL;"
Print #1, "use IEEE.STD_LOGIC_ARITH.ALL;"
Print #1, "use IEEE.STD_LOGIC_UNSIGNED.ALL;"
Print #1, "use IEEE.NUMERIC_STD.ALL;"
Print #1, "library UNISIM;"
Print #1, "use UNISIM.VCOMPONENTS.ALL;"
Print #1,
Print #1, "entity RAM is"
Print #1, "     Port ("
Print #1, "           clk_6M_A       :   in  STD_LOGIC             ;"
Print #1, "           clk_6M_B       :   in  STD_LOGIC             ;"
Print #1, "           adr_RAM_A      :   in  STD_LOGIC_VECTOR (8 downto 0)   ;"
Print #1, "           adr_RAM_B      :   in  STD_LOGIC_VECTOR (8 downto 0)   ;"
Print #1, "           DATA_IN_A      :   in  STD_LOGIC_VECTOR (7 downto 0)   ;"
Print #1, "           DATA_IN_B      :   in  STD_LOGIC_VECTOR (7 downto 0)   ;"
Print #1, "           ENABLE_A       :   in  STD_LOGIC              ;"
Print #1, "           ENABLE_B       :   in  STD_LOGIC              ;"
Print #1, "           WRITE_EN_A  :   in  STD_LOGIC             ;"
Print #1, "           WRITE_EN_B  :   in  STD_LOGIC             ;"
Print #1, "           SET_RESET_A :   in  STD_LOGIC             ;"
Print #1, "           SET_RESET_B :   in  STD_LOGIC             ;"
Print #1, "           DATA_OUT_A  :   out STD_LOGIC_VECTOR (8 downto 1)  ;"
Print #1, "           DATA_OUT_B  :   out STD_LOGIC_VECTOR (8 downto 1)"
Print #1, "          );"
Print #1, "end RAM ;"
Print #1,
Print #1,
Print #1,
Print #1, "architecture RAM_arch of RAM is"
Print #1,
Print #1,
Print #1, "----- Component RAMB4_S8_S8 -----"
Print #1, "component RAMB4_S8_S8"
```

```
Print #1,
Print #1,
Print #1, "  generic ("
Print #1, "      -- mode acquisition"
Print #1, "              -- @ decimale   313029282726252423222120191817161514131211109 8 7 6 5 4 3 2 1 0"
Print #1, "      INIT_00 : bit_vector := X" & Chr(34) & "000000003F3F0000000000" & Chr(34) & " " & Chr(38) & " " & Chr(34) & adr_transposee(3) & Chr(34) & " " & Chr(38) & " " & Chr(34) & adr_transposee(2) & Chr(34) & " " & Chr(38) & " " & Chr(34) & adr_transposee(1) & Chr(34) & " " & Chr(38) & " " & "X" & Chr(34) & "00000000000000000000000000000000000000" & Chr(34) & "; --RAMA 0 à 255"
Print #1, "              -- @ decimale 6362616059585756555453525150494847464544434241403938373635343332"
Print #1, "      INIT_01 : bit_vector := X" & Chr(34) & "000000000000000003F0000000000" & Chr(34); " " & Chr(38) & " " & Chr(34) & adr_transposee(3) & Chr(34) & " " & Chr(38) & " " & Chr(34) & adr_transposee(2) & Chr(34) & " " & Chr(38) & " " & Chr(34) & adr_transposee(1) & Chr(34) & " " & Chr(38) & " " & "X" & Chr(34) & "000000003F000000000000" & Chr(34) & " " & Chr(38) & " " & Chr(34) & adr_transposee(3) & Chr(34) & " " & Chr(38) & " " & Chr(34) & adr_transposee(2) & Chr(34) & " " & Chr(38) & " " & Chr(34) & adr_transposee(1) & Chr(34) & "; --RAMA 256 à 511 "
Print #1, "      INIT_02 : bit_vector := X" & Chr(34) & "0000000000000000000000000000000000000000000000000000000000000000000000" & Chr(34) & "; --RAMA 512 à 767"
Print #1, "      INIT_03 : bit_vector := X" & Chr(34) & "0000000000000000000000000000000000000000000000000000000000000000000000" & Chr(34) & "; --RAMA 768 à 1023"
Print #1, "      INIT_04 : bit_vector := X" & Chr(34) & "0000000000000000000000000000000000000000000000000000000000000000000000" & Chr(34) & "; --RAMA 1024 à 1278"
Print #1, "      INIT_05 : bit_vector := X" & Chr(34) & "0000000000000000000000000000000000000000000000000000000000000000000000" & Chr(34) & "; --RAMA 1279 à 1533"
Print #1, "      INIT_06 : bit_vector := X" & Chr(34) & "0000000000000000000000000000000000000000000000000000000000000000000000" & Chr(34) & "; --RAMA 1534 à ..."
Print #1, "      INIT_07 : bit_vector := X" & Chr(34) & "0000000000000000000000000000000000000000000000000000000000000000000000" & Chr(34) & "; --RAMA"
Print #1,
Print #1, "      -- mode identification"
Print #1, "      INIT_08 : bit_vector := X" & Chr(34) & "0000000000000000000000000000000000000000000000000000000000000000000000" & Chr(34) & "; --RAMB 2048 à 2303"
Print #1, "      INIT_09 : bit_vector := X" & Chr(34) & "00000000000000000000000000" & Chr(34) & " " & Chr(38) & " " & Chr(34) & adr_transposee(3) & Chr(34) & " " & Chr(38) & " " & Chr(34) & adr_transposee(2) & Chr(34) & " " & Chr(38) & " " & Chr(34) & adr_transposee(1) & Chr(34) & " " & Chr(38) &
```

```
" " & "X" & Chr(34) & "00000000000000000000003F3F3F" & Chr(34) & "; --RAMB 2304
à 2559"
Print #1, "      INIT_0A : bit_vector := X" & Chr(34) &
"0000000000000000000000000000000000000000000000000000000000000000" & Chr(34)
& "; --RAMB 2600 à 2815"
Print #1, "      INIT_0B : bit_vector := X" & Chr(34) &
"0000000000000000000000000000000000000000000000000000000000000000" & Chr(34)
& "; --RAMB 2816 à ..."
Print #1, "      INIT_0C : bit_vector := X" & Chr(34) &
"0000000000000000000000000000000000000000000000000000000000000000" & Chr(34)
& "; --RAMB"
Print #1, "      INIT_0D : bit_vector := X" & Chr(34) &
"0000000000000000000000000000000000000000000000000000000000000000" & Chr(34)
& "; --RAMB"
Print #1, "      INIT_0E : bit_vector := X" & Chr(34) &
"0000000000000000000000000000000000000000000000000000000000000000" & Chr(34)
& "; --RAMB"
Print #1, "      INIT_0F : bit_vector := X" & Chr(34) &
"0000000000000000000000000000000000000000000000000000000000000000" & Chr(34)
& "  --RAMB 3840 à 4095"
Print #1, "         );"
Print #1,
Print #1, "port ("
Print #1, "       DIA    : in STD_LOGIC_VECTOR (7 downto 0);"
Print #1, "       DIB    : in STD_LOGIC_VECTOR (7 downto 0);"
Print #1, "       ENA    : in STD_logic;"
Print #1, "       ENB    : in STD_logic;"
Print #1, "       WEA    : in STD_logic;"
Print #1, "       WEB    : in STD_logic;"
Print #1, "       RSTA   : in STD_logic;"
Print #1, "       RSTB   : in STD_logic;"
Print #1, "       CLKA   : in STD_logic;"
Print #1, "       CLKB   : in STD_logic;"
Print #1, "       ADDRA  : in STD_LOGIC_VECTOR (8 downto 0);"
Print #1, "       ADDRB  : in STD_LOGIC_VECTOR (8 downto 0);"
Print #1, "       DOA    : out STD_LOGIC_VECTOR (7 downto 0);"
Print #1, "       DOB    : out STD_LOGIC_VECTOR (7 downto 0)"
Print #1, "      );"
Print #1, "end component;"
Print #1,
Print #1,
Print #1,
Print #1, "begin"
Print #1,
Print #1, "-- bloc ram instantiation"
Print #1, "U_RAMB4_S8_S8:  RAMB4_S8_S8"
Print #1, "port map ("
Print #1, "       CLKA    =>  clk_6M_A    ,"
Print #1, "       CLKB    =>  clk_6M_B    ,"
```

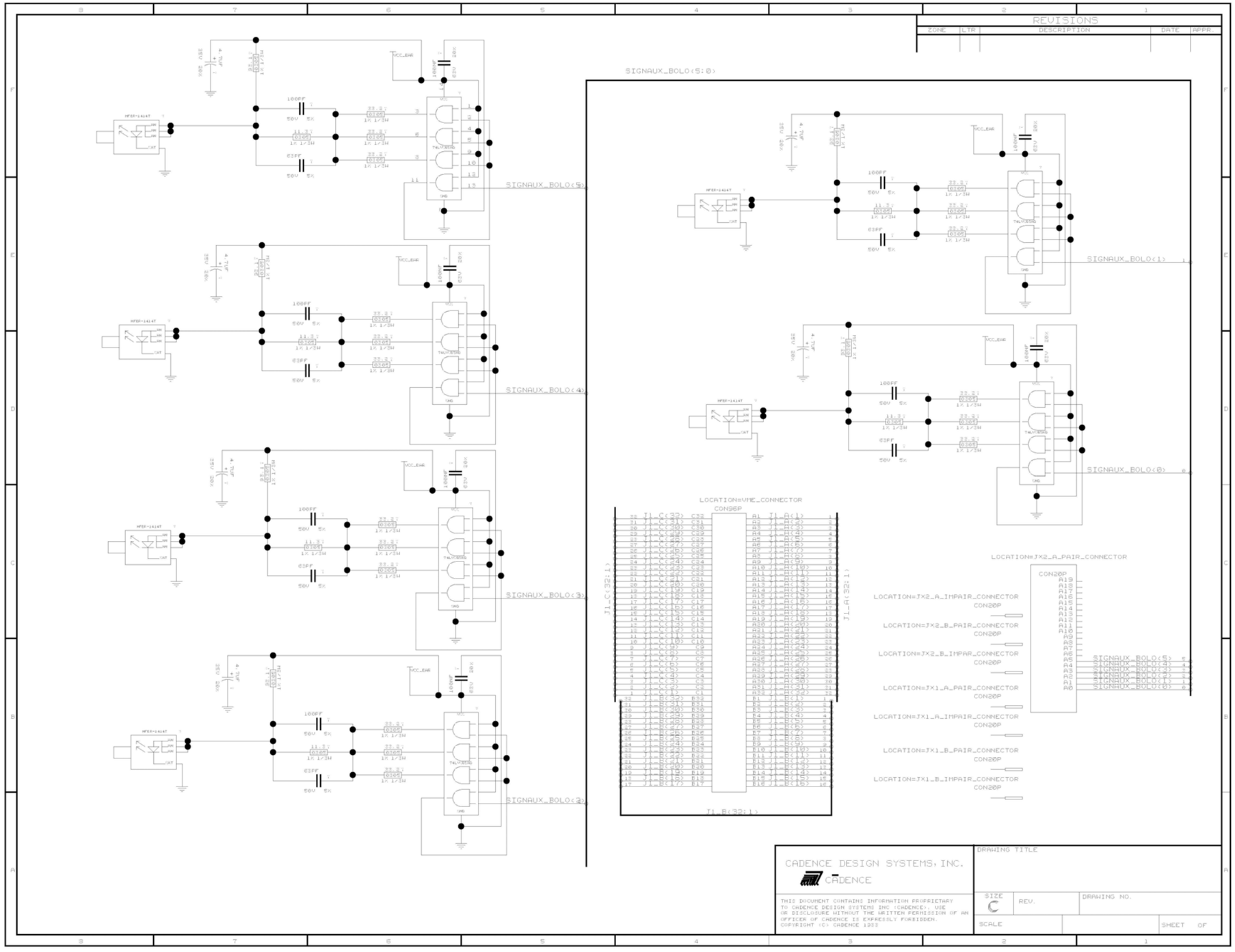
```
Print #1, "       ADDRA   =>  adr_RAM_A ,"
Print #1, "       ADDRB   =>  adr_RAM_B ,"
Print #1, "       DIA     =>  DATA_IN_A ,"
Print #1, "       DIB     =>  DATA_IN_B ,"
Print #1, "       ENA     =>  ENABLE_A    ,"
Print #1, "       ENB     =>  ENABLE_B    ,"
Print #1, "       WEA     =>  WRITE_EN_A ,"
Print #1, "       WEB     =>  WRITE_EN_B ,"
Print #1, "       RSTA    =>  SET_RESET_A,"
Print #1, "       RSTB    =>  SET_RESET_B,"
Print #1, "       DOA     =>  DATA_OUT_A(8 downto 1) ,"
Print #1, "       DOB     =>  DATA_OUT_B(8 downto 1)"
Print #1, "       );"
Print #1,
Print #1, "end;"

Close #1   ' Ferme le fichier RAM.vhd



'------------------- crée le fichier VHDL modélisant les bolo par rapport à leurs adresses ---------
Open "bolo_selectionnes.vhd" For Output As #2   ' Ouvre le fichier en écriture.
Print #2, "library IEEE;"
Print #2, "use IEEE.STD_LOGIC_1164.ALL;"
Print #2, "use IEEE.STD_LOGIC_ARITH.ALL;"
Print #2, "use IEEE.STD_LOGIC_UNSIGNED.ALL;"
Print #2,
Print #2,
Print #2,
Print #2, "entity bolo_selectionnes is"
Print #2, "    Port ("
Print #2, "           synchro               :   in  std_logic := '0'              ;"
Print #2, "           sel_adr_bolo          :   in  STD_LOGIC_VECTOR (8 downto 1)   ;"
Print #2, "           bolo_selectionnes_numero:   out STD_LOGIC_VECTOR (6 downto 1) := "
& Chr(34) & "000000" & Chr(34)
Print #2, "         );"
Print #2, "end bolo_selectionnes;"
Print #2,
Print #2,
Print #2,
Print #2, "architecture bolo_selectionnes_arch of bolo_selectionnes is"
Print #2,
Print #2, "type matrice is array(1 to 6) of std_logic_vector(8 downto 1);"
Print #2, "signal adr_bolo_numero : matrice ;"
Print #2, "signal sel_adr_bolo_integer : integer ;"
Print #2,
Print #2, "begin"
Print #2,
```

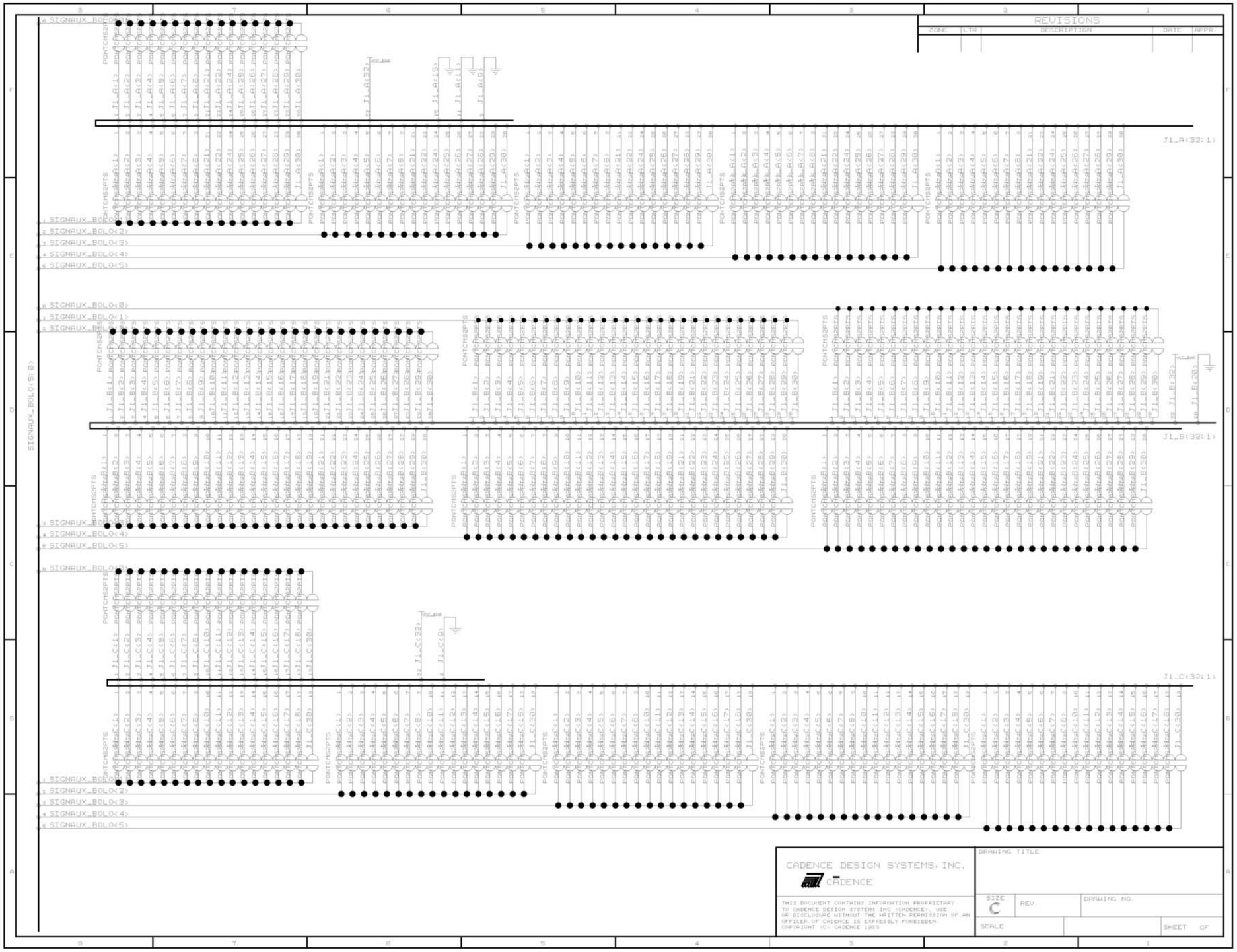
```
Print #2, "-- affectation des adresses correspondantes à chaques bolo"
Print #2, "adr_bolo_numero(1) <= " & Chr(34) & bolo(1) & Chr(34) & ";"
Print #2, "adr_bolo_numero(2) <= " & Chr(34) & bolo(2) & Chr(34) & ";"
Print #2, "adr_bolo_numero(3) <= " & Chr(34) & bolo(3) & Chr(34) & ";"
Print #2, "adr_bolo_numero(4) <= " & Chr(34) & bolo(4) & Chr(34) & ";"
Print #2, "adr_bolo_numero(5) <= " & Chr(34) & bolo(5) & Chr(34) & ";"
Print #2, "adr_bolo_numero(6) <= " & Chr(34) & bolo(6) & Chr(34) & ";"
Print #2,
Print #2,
Print #2, "process (synchro)"
Print #2, "begin"
Print #2, ""
Print #2, "    for i in 1 to 6 loop"
Print #2, "        If sel_adr_bolo_integer = conv_integer(adr_bolo_numero(i)) Then"
Print #2, "            bolo_selectionnes_numero(i) <= '1' ;"
Print #2, "        else bolo_selectionnes_numero(i) <= '0' ;"
Print #2, "        end if;"
Print #2, "    end loop;"
Print #2,
Print #2, "end process;"
Print #2,
Print #2, "sel_adr_bolo_integer <= conv_integer(sel_adr_bolo) ;"
Print #2,
Print #2, "end;"

Close #2   ' Ferme le fichier bolo_selectionnes.vhd

End Sub

Private Sub Command2_Click()       '---- bouton RAZ
Dim i, j As Integer

'------------------------- affecte pour chaque N° de bolo son adresse correspondante ------------
For j = 1 To 6
   matrice(1, j) = j And 1
   matrice(2, j) = (j And 2) / 2
   matrice(3, j) = (j And 4) / 4
   matrice(4, j) = (j And 8) / 8
   matrice(5, j) = (j And 16) / 16
   matrice(6, j) = (j And 32) / 32
   matrice(7, j) = (j And 64) / 64
   matrice(8, j) = (j And 128) / 128
Next j

Text1.Text = "0"
Text2.Text = "0"

'----------------------- constitue les mots de 8 bits pour remplir la RAM ----------------------
For i = 1 To 8
```

```
   adr_transposee(i) = (matrice(i, 8) & matrice(i, 7) & matrice(i, 6) & matrice(i, 5) &
matrice(i, 4) & matrice(i, 3) & matrice(i, 2) & matrice(i, 1))
Next i

End Sub

Private Sub Command3_Click()       '---- bouton valider
Dim i As Integer

'----------------------- enregistre une adresse particulière d'un bolo particulier ----------
If Text1 < 9 And Text1 > 0 And Text2 > 0 And Text2 < 121 Then
   matrice(1, Text1.Text) = Text2.Text And 1
   matrice(2, Text1.Text) = (Text2.Text And 2) / 2 ' "And 2) / 2" sert à ne récupérer
l'information 1 ou 0 du 2ième bit
   matrice(3, Text1.Text) = (Text2.Text And 4) / 4
   matrice(4, Text1.Text) = (Text2.Text And 8) / 8
   matrice(5, Text1.Text) = (Text2.Text And 16) / 16
   matrice(6, Text1.Text) = (Text2.Text And 32) / 32
   matrice(7, Text1.Text) = (Text2.Text And 64) / 64
   matrice(8, Text1.Text) = (Text2.Text And 128) / 128
Else
   Form3.Visible = True
End If

Text1.Text = "0"
Text2.Text = "0"

'------------------------ constitue les mots de 8 bits pour remplir la RAM ----------------------
For i = 1 To 8
   adr_transposee(i) = (matrice(i, 8) & matrice(i, 7) & matrice(i, 6) & matrice(i, 5) &
matrice(i, 4) & matrice(i, 3) & matrice(i, 2) & matrice(i, 1))
Next i

End Sub

Private Sub Form_Load()      '---- à l'ouverture de la fenêtre principale

Dim i, j As Integer

'------------------------- affecte pour chaque N° de bolo son adresse correspondante ------------
For j = 1 To 6
   matrice(1, j) = j And 1
   matrice(2, j) = (j And 2) / 2
   matrice(3, j) = (j And 4) / 4
   matrice(4, j) = (j And 8) / 8
   matrice(5, j) = (j And 16) / 16
   matrice(6, j) = (j And 32) / 32
   matrice(7, j) = (j And 64) / 64
   matrice(8, j) = (j And 128) / 128
```

```
Next j

Text1.Text = "0"
Text2.Text = "0"

'------------------------ constitue les mots de 8 bits pour remplir la RAM ----------------------
For i = 1 To 8
   adr_transposee(i) = (matrice(i, 8) & matrice(i, 7) & matrice(i, 6) & matrice(i, 5) &
matrice(i, 4) & matrice(i, 3) & matrice(i, 2) & matrice(i, 1))
Next i

End Sub

Private Sub help_Click()       '---- ouverture de la fenêtre d'aide
Form2.Visible = True
End Sub

Private Sub Save_Click()       '---- ouvre la fenêtre d'exploration save as…
CommonDialog1.ShowSave
End Sub

Private Sub Timer1_Timer()
Dim i, j As Integer

'------------------------ affecte à tt moment l'adresse correspondante au bolo ... --------------
For i = 1 To 8
   bolo(i) = matrice(8, i) & matrice(7, i) & matrice(6, i) & matrice(5, i) & matrice(4, i) &
matrice(3, i) & matrice(2, i) & matrice(1, i)
Next i

'----------------------- ... et affiche l'état courant ----------------------------------------
Label1 = "Adresses" & Chr(13) & Chr(13) & bolo(1) & Chr(13) & bolo(2) & Chr(13) &
bolo(3) & Chr(13) & bolo(4) & Chr(13) & bolo(5) & Chr(13) & bolo(6) & Chr(13) & bolo(7)
& Chr(13) & bolo(8) & Chr(13) & "..."

End Sub
```

Annexe 6 : Schéma de la carte EFEBB




REVISIONS
ZONE
LTR
DESCRIPTION
DATE
APPR.
SIGNAUX_BOLO<5:0>
J1_A<32:1>
J1_B<32:1>
J1_C<32:1>
CADENCE DESIGN SYSTEMS, INC.
CADENCE
DRAWING TITLE
SIZE
C
REV.
DRAWING NO.
SCALE
SHEET OF