Une architecture programmable de traitement des impulsions pour l'instrumentation nucléaire

YOANN MOLINE[1], MATHIEU THEVENIN[1], GWENOLE CORRE[1], MICHEL PAINDAVOINE[2]

[1] CEA, LIST - Laboratoire Capteurs et Architectures Électroniques, F-91191 Gif-sur-Yvette, France
[2] CNRS, Université de Bourgogne - Laboratoire d'Étude de l'Apprentissage et du Développement, 21000 DIJON, France
[1]{prenom.nom}@cea.fr, [2]paindav@u-bourgogne.fr

Résumé - Cet article présente une architecture de type Multiple Program Multiple Data (MPMD), développée pour répondre aux contraintes liées à l'instrumentation nucléaire : temps réel, flexibilité pour le multivoie, gestion du temps mort et programmabilité. Cette architecture est composée d'un ensemble d'unités fonctionnelles (FU) programmables indépendantes pilotées par les impulsions issues du signal de mesure de la radioactivité. Ces FUs sont capables de gérer l'arrivée d'évènements non déterministes et des durées d'exécution de programme variables. Le prototype virtuel de cette architecture est développé en SystemC au cycle d'horloge près, et donne des résultats prometteurs en termes de passage à l'échelle tout en maintenant le zéro temps mort. Cette architecture ouvre la voie à des applications innovantes de traitement numérique des impulsions jusqu'alors limitées aux traitements hors-ligne.

Summary - This paper introduces a specific Multiple Program Multiple Data (MPMD) architecture particularly suited to address the issues of Nuclear Instrumentation: real time, multichannel flexibility, dead-time management and programmability. This architecture is composed of a set of independent and programmable Functional Units (FUs) which execution is driven by the pulses which compose the radioactivity measurements. They are able to manage non-deterministic events and program durations. The virtual prototype of the architecture is developed in cycle accurate SystemC and shows promising results in terms of scalability while maintaining zero dead-time. This architecture paves the way for novel embedded real time pulse processing restricted until now to offline processing.

1 Introduction

Le domaine de l'instrumentation nucléaire couvre un large éventail d'applications comme le comptage, la spectrométrie, la discrimination n-γ ou la coïncidence temporelle. De nouveaux algorithmes pour ces applications sont constamment proposés grâce aux avancées dans le domaine du traitement du signal. Cependant, ils ne sont pas encore mis en œuvre dans les systèmes de mesure actuels qui doivent faire face à deux problématiques majeures. La première est caractéristique au signal lui-même, qui est constitué d'impulsions dont les amplitudes, les durées, et les dates d'arrivée sont aléatoires puisque issues d'un processus poissonnien homogène [1]. La seconde est l'exigence temps réel, impliquant la paralysie des chaînes de mesure après l'arrivée d'un événement. Aucun nouvel événement ne peut être traité par le système durant cette paralysie. Ce phénomène, appelé temps mort, est dû aux limitations de l'électronique de traitement du signal qui doit terminer ses calculs avant d'en accepter d'autres. Les applications du domaine médical ou liées à la sécurité doivent limiter le temps mort afin d'exploiter un maximum d'informations en un minimum de temps. Elles utilisent actuellement des composants reconfigurables de type FPGA [2]. Cependant, l'implémentation d'algorithmes dédiés sur les technologies reconfigurables est une tâche complexe et coûteuse en temps. Pour toutes ces raisons, une architecture numérique de traitement des impulsions pouvant être programmée dans un langage de haut niveau tel que le C ou le C++ est nécessaire. Toutefois, les performances de solutions programmables actuelles ne permettent pas un fonctionnement en ligne sans augmentation de temps mort. Cette problématique augmente sensiblement avec le nombre de voies d'acquisition qui peut dépasser la centaine de voies [3].

Cet article présente une architecture de traitement des impulsions asynchrone de type Multiple Program Multiple Data (MPMD) capable de répondre à ces contraintes. Son modèle d'exécution se base sur les caractéristiques non-déterministes du signal. Elle répond à la problématique du temps mort tout en étant programmable et flexible avec l'évolution du nombre de voies d'acquisition.

2 État de l'art

Les architectures électroniques utilisées dans l'instrumentation nucléaire doivent répondre aux besoins de flexibilité, de passage à l'échelle en multivoie et de gestion du temps mort, sans pour autant être dimensionnées pour le pire cas.

Un premier système programmable tente de répondre à la problématique de temps mort [4]. L'architecture, utilisée pour la spectrométrie, utilise un processeur de traitement numérique du signal de type DSP associé à une mémoire hiérarchisée en deux niveaux. Cette architecture est améliorée en [5] avec l'utilisation de plusieurs tuiles de calcul, chacune comprenant un DSP et un second niveau de mémoire dédiés. Si un nombre suffisant de tuiles de calcul est présent, le temps mort tend vers zéro. Néanmoins, cette approche n'est pas en mesure de gérer plusieurs voies de mesure. Des travaux plus récents [6] proposent une architecture reconfigurable sur FPGA. Cette architecture introduit la notion de macro-pipeline, séparant les étages de traitement, chacun comprenant un algorithme de traitement d'impulsion. Cela permet de réduire le temps mort à la latence de l'étage le plus lent. L'architecture présentée en [2] est la première capable de réaliser aussi

bien du comptage, de la spectrométrie gamma, de la discrimination neutron-gamma (n-γ), et de la corrélation temporelle. Le système utilise quatre voies d'acquisition. Chaque voie possède sa propre tuile de calcul. Les traitements associés à chaque tuile sont implémentés indépendamment. Ils peuvent fonctionner en parallèle sur le signal afin de réduire le temps mort. Cette architecture est la plus proche d'une plateforme multi-applicative mais reste dépendante de *firmwares* dédiés et limitée à quatre voies de mesure. La dernière architecture présentée dans [7] sépare les impulsions du reste du signal avant tout autre traitement. Les traitements ultérieurs travaillent donc uniquement sur les impulsions, ce qui réduit les besoins en ressources de calcul et donc le temps mort. Ces impulsions sont distribuées sur deux niveaux de calcul parallèles. Cependant, l'étape de *trigg* utilisée pour extraire les impulsions n'est pas capable de s'adapter aux longueurs variables des impulsions, et beaucoup d'entre elles sont rejetées de par l'utilisation d'une fenêtre d'observation statique. Cette solution n'offre pas la flexibilité proposée par [2], ni la gestion du temps mort de [5] et n'est pas programmable ni multivoie. Individuellement, les travaux présents dans la littérature ne répondent pas conjointement aux problématiques de suppression du temps mort, de flexibilité ou de multivoies.

3 Proposition architecturale

3.1 Extraction des impulsions en amont

La première étape de notre modèle est l'extraction des impulsions. Comme le montrent les travaux présentés dans [7] et [8], l'étape de *trigg* peut être réalisée directement sur le signal brut après la conversion analogique-numérique. La datation précise des impulsions et la détection des empilements [8] sont réalisées à cette étape. Cet étage produit donc des paquets contenant les impulsions individuelles associées à leurs dates d'arrivée. Cela permet à chaque application en aval de travailler sur des tableaux de données de tailles variables simplifiant l'implémentation des algorithmes associés. Le nombre de paquet à distribuer varie avec le volume d'impulsion.

3.2 Modèle d'exécution dirigé par les impulsions

Les algorithmes de DPP traitent les impulsions individuellement (discrimination n-γ [9], spectrométrie [1]) avant de fusionner les informations extraites par un traitement final (corrélation temporelle, construction d'un histogramme).

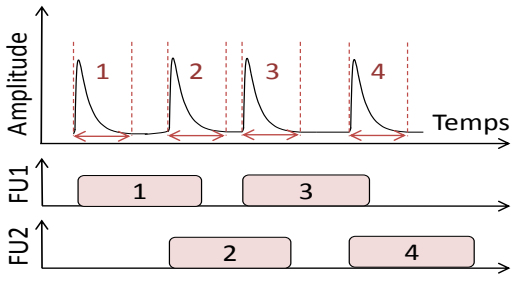

Fig. 1. Distribution des impulsions sur différentes ressources du calcul. Dans cet exemple, toutes les impulsions sont traitées.

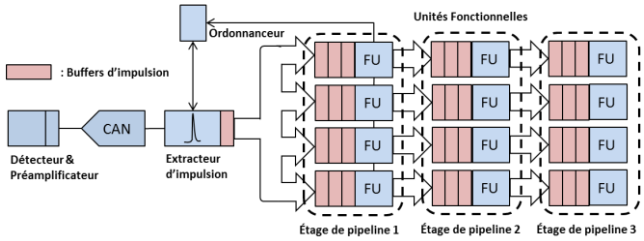

Fig. 2. Modèle d'exécution asynchrone macro-pipeliné et dirigé par les impulsions.

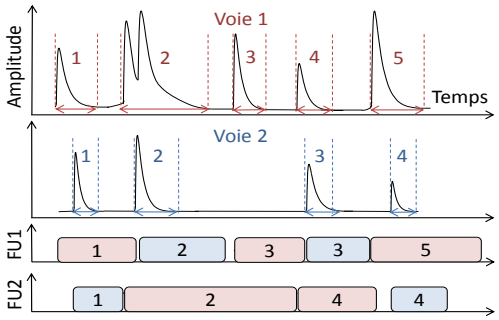

Fig. 3. Affectation du traitement des impulsions aux FUs sur une architecture multivoie à ressources partagées.

Une fois les impulsions extraites, il est donc possible de les distribuer individuellement sur des unités fonctionnelles (FUs) à la manière de [5]. La Fig. 1 illustre un chronogramme représentant les durées d'occupation des FUs corrélées à l'arrivée d'impulsions. Dans cet exemple, la FU1 seule ne permet pas de traiter l'intégralité des impulsions à cause de la durée de son traitement. Dans cet exemple, un système sans pertes est possible par l'ajout d'une seconde FU si les impulsions sont correctement distribuées. Pour cela, un ordonnanceur (première-FU-disponible, première-FU-servie) et un réseau d'interconnexion sont utilisés. La date d'arrivée des impulsions étant connue, elles peuvent être traitées individuellement et dans le désordre sans contrainte de synchronisation. Cela conduit à un modèle d'exécution asynchrone dont les traitements sont « dirigés par les impulsions ».

3.3 Macro-pipeline logiciel et matériel

Une chaîne de mesure est composée d'une succession d'algorithmes. Ils forment naturellement un macro-pipeline logiciel tel que dans [6]. Comme l'illustre la Fig. 2, notre approche consiste à assigner un algorithme par FU afin de réduire la latence globale à la pire latence d'exécution de FU du pipeline. Comme chaque traitement est exécuté localement et de manière indépendante sur une FU, cette approche limite les ressources requises en mémoire partagée. Elle est également entièrement compatible avec notre modèle asynchrone dirigé par les impulsions puisque chaque FU distribue des paquets de taille variable dès que possible sur la première FU disponible quelle que soit la durée du traitement.

3.4 Partage des ressources pour le multivoie

Augmenter le nombre de voies implique que plusieurs extracteurs d'impulsion doivent distribuer leurs impulsions sur les FUs disponibles. De ce fait, plus d'impulsions doivent être traitées, augmentant ainsi

le nombre de FUs requises. Cependant, chaque voie est indépendante et reçoit des évènements de manière aléatoire, ce qui veut dire que les extracteurs d'impulsion associés n'ont pas forcément besoin d'accéder simultanément à leurs FUs pour distribuer leurs impulsions. Par conséquent, partager les FUs entre les voies d'acquisition peut réduire le délai pour trouver une FU disponible. Dans l'exemple illustré par la Fig. 3, seules deux FUs sont requises pour traiter les impulsions issues de deux voies d'acquisition. Sans le partage des FUs, la voie 1 nécessiterait deux FUs à elle seule. La durée des traitements d'une FU est dépendante de l'algorithme qui y est implémenté et de la durée de l'impulsion elle-même, le partage de ressources peut donc être effectué entre chaque étage de pipeline. Par conséquent chaque étage peut être hétérogène en termes de nombre de FU et de capacité de traitement. Un commutateur parfait (*e.g crossbar*) est utilisé entre chaque étage de FU. Il est piloté par l'ordonnanceur et est limité à une commande de routage par cycle. Le modèle final combinant tous les éléments décrits précédemment est présenté dans la Fig. 4.

3.5 Unités fonctionnelles (FUs)

Les FUs sont composées de processeurs génériques pour l'exécution des algorithmes et d'une partie contrôle s'occupant de la transmission des données. Le changement d'une application n'implique pas de modifications coûteuses du matériel mais uniquement une recompilation. Chaque FU étant indépendante, avec son propre domaine d'horloge, une hiérarchie mémoire spécifique doit être conçue pour assurer l'entrée et la sortie des données avec le reste de l'architecture. Une FU est présentée dans la Fig. 5. Deux FIFOs à double horloge sont utilisées pour la communication entre les deux domaines d'horloge. Leurs tailles sont dimensionnées pour contenir la longueur maximum d'une impulsion ou d'un empilement toléré par l'extracteur d'impulsion, évitant ainsi tout débordement mémoire. Une seule impulsion peut être traitée à la fois par une FU afin d'éviter la gestion d'impulsion tronquée. Dans ce cas, la FU est considérée comme occupée (*busy*) et ne peut pas être choisie par l'ordonnanceur pour recevoir une impulsion. L'exécution des traitements est déclenchée par la présence d'une impulsion/paquet grâce aux signaux de contrôle *FIFO Empty*. Le code du processeur est modélisé pour ajouter différents types de métadonnées en fonction des applications.

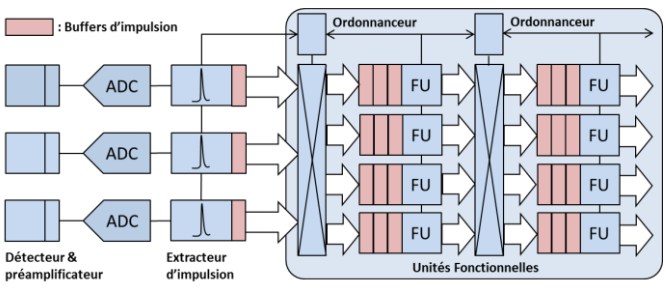

Fig. 4. Modèle d'architecture proposé pour répondre aux contraintes du domaine de l'instrumentation nucléaire.

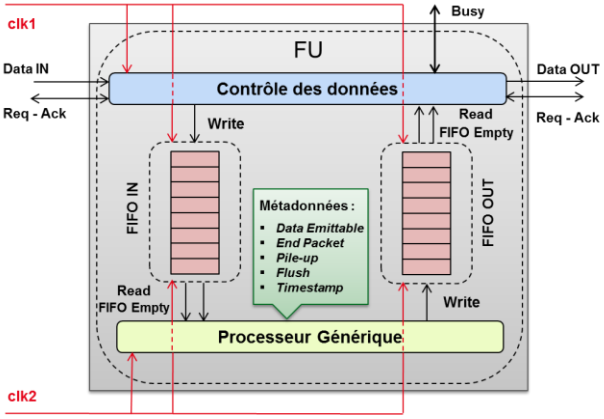

Fig. 5. Modèle de FU composée de deux FIFOs à double horloge.

Par exemple, *Data Emittable* signale au contrôleur la possibilité de transmettre en flux des données pour des algorithmes pipelinable (filtres), tandis qu'*End Packet* signale la mémorisation complète de l'impulsion/paquet dans la FIFO de sortie avant sa transmission (vérification d'une impulsion avant envoi).

4 Résultats expérimentaux

Le macro-modèle de l'architecture est développé en SystemC au cycle d'horloge près. Il permet des modifications aisées de l'architecture (nombre de voies, nombre de FUs), cela permet d'implémenter différents algorithmes et tester différents types de réseaux d'interconnexion, ordonnanceurs et modèle d'exécution.

4.1 Protocole expérimental

Il n'existe aucun benchmark spécifique comparant des architectures de traitement d'impulsions entre elles dans la littérature. Cependant, les travaux présentés dans [10] utilisent un simulateur dédié pour évaluer l'évolution du temps mort en fonction d'une application donnée et du nombre de tuiles de calcul utilisées. En partant de cette idée, nous proposons un benchmark basé sur le nombre d'impulsions perdues à cause du manque de ressources (mémoire, capacité de calcul etc.) par un système de traitement d'impulsions, pour un signal donné. Cela rend possible une comparaison équitable entre architectures à condition que le même jeu de données et extracteur d'impulsion soient utilisés. Les données utilisées pour les tests ont été obtenues à l'aide d'une source Am-241 et d'un détecteur 4πγ NaI(TI)-puit. La conversion analogique numérique est résolue sur 14-bit et cadencée à 125 MHz. Le signal est ensuite mémorisé dans des fichiers correspondant à deux secondes de mesure. Ils contiennent statistiquement plus de 17 000 impulsions au regard de l'activité de la source. Chaque fichier est alors associé à une voie de mesure modélisée en SystemC. Les FUs étant programmables, la modification du logiciel impacte directement la durée d'exécution du programme. Afin d'être représentatif d'un grand nombre d'applications, l'analyse exhaustive des algorithmes traditionnels de traitement des impulsions est réalisée. Enfin, la charge des algorithmes est traduite par le nombre de cycles requis pour traiter un échantillon d'une impulsion.

4.2 Présentation des résultats

4.2.1 Simulation de la distribution des impulsions

Afin d'évaluer le nombre de FUs requis pour atteindre le zéro temps mort, nous nous proposons d'exécuter plusieurs instances de l'architecture pour une seule voie d'acquisition. Plusieurs exécutions du modèle sont réalisées en utilisant le même jeu de données, mais pour différents nombres de FUs et durées d'exécution du programme. Enfin, le nombre d'impulsions perdues par manque de ressources est présenté en Fig. 6.

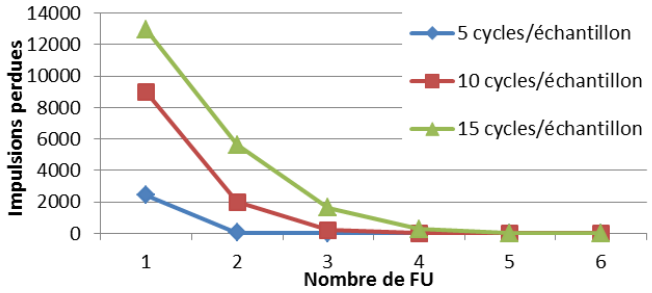

Fig. 6. Résultats de simulation pour une voie d'acquisition relatifs au nombre de FUs nécessaires au zéro temps mort.

Ces résultats montrent que la distribution des impulsions sur les FUs permet d'atteindre le zéro temps mort, confirmant ainsi la théorie. Le gain obtenu par l'ajout de FUs décroit exponentiellement comme démontré dans [11]. Dans notre exemple, aucune impulsion n'est perdue avec six FUs, quelle que soit la configuration du simulateur. Ces résultats montrent également qu'un compromis peut être trouvé pour se rapprocher du zéro temps mort sans sur-dimensionner notre architecture. Cela est confirmé par l'analyse du temps d'occupation d'une FU (obtenu à partir de l'état *busy*) durant une acquisition comme présenté dans le Tab. 1. A titre d'exemple, la sixième FU n'est occupée que 0.34% du temps, son utilité peut donc être remise en question dans certaines applications.

4.3 Simulation du partage de ressources

Afin d'évaluer le gain, en termes de nombre de FUs, obtenu grâce à notre proposition de partage de FUs entre voies d'acquisition, nous proposons d'exécuter une simulation sur une instance de l'architecture comprenant deux voies de mesures acquérant simultanément du signal. Le nombre d'impulsions perdues est ainsi enregistré pour la voie 1 uniquement (voie 2 désactivée), pour la voie 2 uniquement (voie 1 désactivée), pour les deux voies sans partage de FUs et pour les deux voies avec le partage de FUs. Les résultats sont présentés en Tab. 2. Pour chaque configuration testée, un nombre significatif de FUs est économisé lorsque le partage des ressources est activé. Le gain décroit plus lentement que la durée d'exécution du programme n'augmente. L'évolution du gain suit une exponentielle décroissante comme dans les résultats précédents. Cela confirme qu'utiliser la caractéristique d'arrivée aléatoire des impulsions sur chaque voie pour partager les FUs permet un meilleur passage à l'échelle de l'architecture pour les applications multivoie.

Tab. I : SIMULATION DU TEMPS D'OCCUPATION DES FUS POUR 15 CYCLES/ECHANTILLON

Nombre de FU	Taux d'occupation d'une FU durant la mesure (en %)					
	FU 1	FU 2	FU 3	FU 4	FU 5	FU 6
1	97.71	N.A	N.A	N.A	N.A	N.A
2	88.43	82.79	N.A	N.A	N.A	N.A
3	81.44	72.78	57.84	N.A	N.A	N.A
4	75.42	68.73	52.28	29.11	N.A	N.A
5	75.20	67.09	49.90	27.19	7.63	N.A
6	75.20	67.09	49.96	26.79	7.63	0.34

Tab. II : SIMULATION DU GAIN EN TERMES DE PASSAGE A L'ECHELLE OFFERT PAR LE PARTAGE DE RESSOURCES

Protocole d'acquisition	Nombre de FU requis			
	15 cycles/éch	10 cycles/éch	5 cycles/éch	2 cycles/éch
Voie 1	6	4	3	2
Voie 2	6	4	3	2
Voie 1&2	12	8	6	4
Voie 1&2, FU partagées	9	6	4	2
Gain (%)	**25**	**25**	**33**	**50**

5 Conclusion

Une architecture innovante asynchrone, dirigée par les impulsions et de type MPMD est proposée. Cette architecture est programmable et particulièrement adaptée au traitement numérique des impulsions pour l'instrumentation nucléaire. La séparation des impulsions et leur distribution sur différents étages pipelinés de FUs programmables résout le problème du temps mort et de la programmabilité. Le passage à l'échelle pour les applications multivoie est grandement amélioré par le partage des FUs entre voies de mesure. Le simulateur développé en SystemC montre des résultats prometteurs en termes de passage à l'échelle tout en maintenant le zéro temps mort.

6 Bibliographie

[1] G. F. Knoll, *Radiation Detection and Measurement*, 4th ed. Wiley, John & Sons, 2010.

[2] R. T. Schiffer, M. Flaska, S. A. Pozzi, S. Carney, and D. D. Wentzloff, "A Scalable FPGA-based Digitizing Platform for Radiation Data Acquisition," *Nucl. Instruments Methods Phys. Res. Sect. A.*, vol. 652, no. 1, pp. 491–493, Oct. 2011.

[3] D. Bazzacco, "The Advanced Gamma Ray Tracking Array AGATA," *Nucl. Instruments Methods Phys. Res. Sect. A.*, vol. 746, pp. 248–254, Dec. 2004.

[4] J. Basilio Simoes and C. M. B. . Correia, "Pulse Processing Architectures," *Nucl. Instruments Methods Phys. Res. Sect. A.*, vol. 422, no. 1–3, pp. 405–410, Feb. 1999.

[5] J. M. . Cardoso, J. Basilio Simoes, and C. M. B. . Correia, "A High Performance Reconfigurable Hardware Platform for Digital Pulse Processing," *Nucl. Sci. IEEE Trans.*, vol. 51, no. 3, pp. 921–925, 2004.

[6] S. Normand, V. Kondrasov, G. Corre, and C. Passard, "PING : A New Approach For Nuclear Fuel Cycle Instrumentation," *1st Int. Conf. Adv. Nucl. Instrumentation, Meas. Methods their Appl.*, pp. 1–4, Jun. 2009.

[7] P. Lee, C. Lee, and J. Lee, "Development of FPGA-based Digital Signal Processing System for Radiation Spectroscopy," *Radiat. Meas.*, vol. 48, pp. 12–17, 2012.

[8] Y. Moline, M. Thevenin, G. Corre, and T. Peyret, "Procédé et système d'extraction dynamique d'impulsions dans un signal temporel bruité," *Pat. number FR14 50568*, 2014.

[9] M. Moszynski, G. Bizard, and G. Costa, "Study of n-γ Discrimination by Digital Charge Comparison Method for a Large Volume Liquid Scintillator," *Nucl. Instruments Methods Phys. Res. Sect. A.*, vol. 317, no. 1–2, pp. 262–272, 1992.

[10] J. M. . Cardoso, J. Basilio Simoes, and C. M. B. a. Correia, "Dead-time Analysis of Digital Spectrometers," *Nucl. Instruments Methods Phys. Res. Sect. A.*, vol. 522, no. 3, pp. 487–494, Apr. 2004.