Approximate Computing et Conception d'Opérateurs Arithmétiques Approximatifs

Mengting Zhao

Approximate Computing et Conception d'Opérateurs Arithmétiques Approximatifs

Mengting Zhao, COMELEC, TELECOM Paristech

RESUME

Étant donné que les dimensions physiques des circuits CMOS sont réduites à quelques dizaines de nanomètres, il a été de plus en plus difficile d'améliorer les performances du circuit et d'améliorer l'efficacité énergétique. La technique de calcul approximatif, à la fois logiciel et matériel, a été considérée comme une nouvelle approche d'économie d'énergie et de la surface, ainsi que l'augmentation des performances, mais au prix de certaine perte de précision. L'idée principale est de troquer une certaine précision pour d'autres gains [1], tels que la puissance, la surface et le retard. En particulier, les additionneurs approximatifs ont été considérés très prometteurs pour les applications pouvant tolérer des erreurs.

1. Introduction

Étant donné que les dimensions physiques des circuits CMOS sont réduites à quelques dizaines de nanomètres, il est devenu de plus en plus difficile d'améliorer les performances du circuit tout en garantissant une efficacité énergétique [1-3]. Considérant la croissance explosive des appareils portables tels que les téléphones intelligents, l'amélioration de l'efficacité énergétique des circuits numériques devient de plus en plus importante. *Approximate Computing*, à la fois logiciel et matériel, a été considérée comme une nouvelle approche permettant de satisfaire des contraintes d'énergie et performance, mais au prix d'une certaine perte de précision. Il existe de nombreuses solutions pour les systèmes informatiques numériques d'embrasser une amélioration significative de la vitesse de calcul et une efficacité énergétique en exploitant des méthodologies de calcul approximatives dans les logiciels et le matériel. Afin de tirer profit efficacement un système de calcul approximatif, il est nécessaire de développer de nouvelles architectures matérielles pouvant mettre en ouvre cette approche. Par conséquent, la recherche sur les circuits arithmétiques est essentielle pour la construction d'une plateforme matérielle de calcul approximative ou d'un processeur à usage général qui prend en charge les opérations approximatives. Au cours de mon stage, je me suis concentrée sur les additionneurs approximatifs qui ont été considérés comme une alternative potentielle pour les applications à tolerance d'erreur. L'idée principale est donc d'échanger une certaine précision par des gains dans d'autres métriques propres des circuits, telles que la puissance, la surface et le retard.Les résumées comme suit:

1) L'étude de méthodes conventionnelles reportées dans la littérature pour obtenir une comprehension complète de Approximate Computing.

2) Une recherche sur les Additionneurs Approximatifs et l'élaboration d'un document

résumant une etude comparative des solutions existantes
3) Synthèses des circuits électroniques et simulations réalisées sous Cadence
4) Proposition d'une nouvelle structure d'additionneur en technologie hybride MOS/MTJ combinée avec l'exploration des propriétés des mémoires STT-MTJ.

2. Préliminaires

2.1. STT MTJ

MTJ (Magnetic Tunnel Junctions) sont des éléments de base de la spintronique. Ils se composent essentiellement de trois couches: une couche mince d'oxyde de tunnel feuilletée entre deux couches ferromagnétiques (FM) [8]. En raison de l'effet TMR (Tunnel Magnetoresistance), la résistance à nanopilier, R_p or R_{AP}, dépend de l'orientation relative de l'aimantation des deux couches FM [4], parallèle (P) ou antiparallèle (AP) comme présenté dans la Figure 2.1.

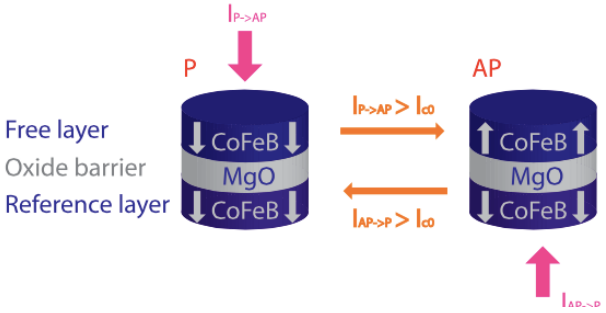


Figure 2.1 Transformation d'état de MTJ

Remarquablement, dans le MgO de base MTJs, le ratio TMR peut atteindre une valeur très élevée (600% par exemple) à la température ambiante, ce qui permet une détection facile de l'état de MTJs par les CMOS amplificateurs. Il est possible de passer d'une configuration à son inverse en utilisant le phénomène STT (Spin-Transfer Torque). Le processus de transformation peut être conclu de la manière suivante: Une transformation de l'état P à l'AP a lieu sous la direction de flux positif d'électrons avec une valeur supérieure au courant critique de commutation I_{c0}. Au contraire, une transformation d'état AP au P requiert un flux négatif d'électrons aussi supérieur au courant critique I_{c0} [5-7]. Les facteurs importants tels que le courant fonctionnel de MTJ (I), le courant critique de commutation et le temps de commutation (T) sont inclus dans le modèle de comportement de MTJ [9]. De plus, en fonction de l'amplitude du courant circulant à travers le MTJ, la durée moyenne de l'opération de commutation peut être calculée par les formules suivantes:

$$\tau = \tau_0 * \exp\left(\frac{E}{k_B T}\left(1-\frac{I}{I_{C0}}\right)\right), \quad when \ I < 0.8 I_{C0}$$

$$\frac{1}{\tau} = \left[\frac{2}{C + \ln\left(\frac{\pi^2 \zeta}{4}\right)}\right] \frac{\mu_B P_{ref}}{e m_m (1+P_{ref} P_{free})} (I - I_{C0}), \quad when \ I > I_{C0}$$

où E est l'énergie de la barrière, t 0 est la période d'essai, T est la t

empérature, kB est la constante de Boltzmann, C est la constante d'Euler, c est le facteur de stabilité thermique, mm est le moment d'aimantation, B est le magnéton de Bohr, Pref et Pfree sont les polarisations du spin tunnel.

Cependant, la commutation STT de MTJ est stochastique intrinsèquement [10], et ce sont les résultats des fluctuations thermiques inévitables de l'aimantation suivant une distribution gaussienne. A cause de ce phénomène, les erreurs d'écriture peuvent se produire. Par exemple, les données souhaitées peuvent ne pas être stockées correctement dans l'opération d'écriture et des commutations inattendues peuvent aussi arriver dans l'opération de détection. Ce phénomène exerce une influence profonde sur la fiabilité des circuits hybrides CMOS/MTJ. La mise en oeuvre de logique dans des mémoires a été proposée pour obtenir efficacité énergétique [11]. La logique peut être réalisée dans une mémoire non volatile à l'aide d'un circuit hybride MOS/MTJ avec un circuit amplificateur de pré-charge de type PCSA. Il est constitué par des dispositifs MTJ, circuit non-volatile d'écriture et arbre logique-circuit. Les opérations d'écriture et de lecture de MTJ sont actives indépendamment par le signal d'horloge. Lorsque le signal d'horloge est élevé, le mode de lecture (détection) sera exécuté. Les fonctions booléennes telles que AND, OR, XOR peuvent être réalisées à travers un arbre logique basé sur des pass transistors. Le transistor PMOS à couplage croisé joue le rôle de gardien pour générer des sorties binaries complémentaires. Le mode d'écriture sera exécuté lorsque le signal d'horloge est bas. Afin de renforcer le courant d'écriture, normalement les ratios W/L de transistors sont augmentés. Pour que les données soient sauvegardées avec succès, un courant minimal d'écriture est nécessaire pour la commutation d'état de MTJ.

2.2. Design de reference – additionneurs conventionnels

Un FA, additionneur complet en français, c'est un circuit avec trois entrées et deux sorties. Il ajoute deux bits d'opérandes d'entrées plus un bit de retenue entrante (Cin) et puis délivre un bit de somme et un bit de retenue sortante (Cout). La somme (S) d'un FA est le résultat de l'opération XOR de tous les trois entrées. Le schématique du FA (Figure 2.2) sont présentées ci-dessous. Le circuit dans la Figure 2.2 est une représentation schématique des deux équations (2) qui expliquent la dépendance des deux sorties S et Carry-out par rapport aux opérandes A, B et la retenue entrante Cin. Il existe plusieurs assemblages de transistors pouvant mettre en oeuvre les équations (2). Par exemple dans la Figure 2.3, un circuit qui s'appelle XA (XOR/XNOR-based

Adder) est composé de dix transistors et un autre design dénommé MA (Mirror Adder) contient vingt-quatre transistors en total.

$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + (A+B)C_{in} \qquad (2)$$

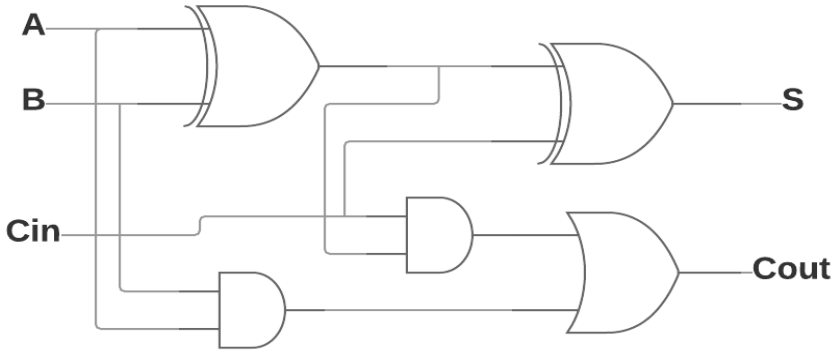


Figure 2.2 Schématique du FA

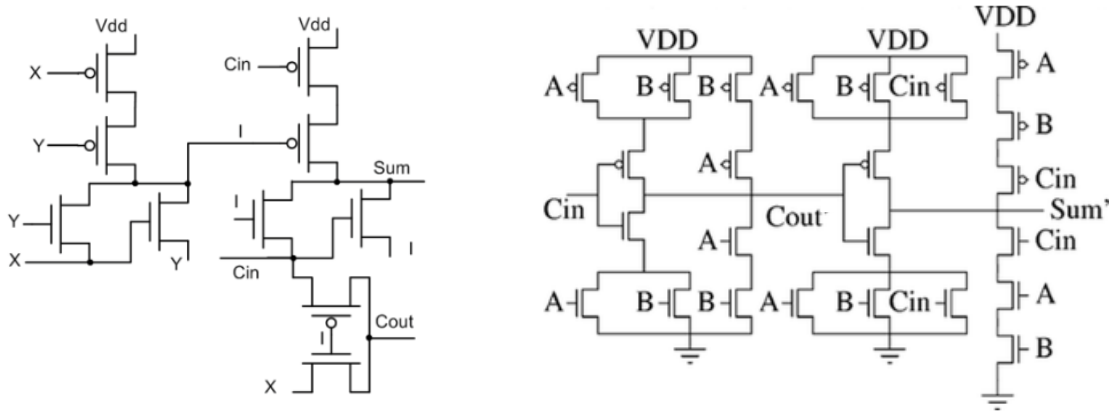


Figure 2.3 (a)Schématique du XA (XOR/XNOR-based Adder) (b)Schématique du MA (Mirror Adder)

Les additionneurs sont utilisés pour le calcul de l'addition de deux nombres binaires. Parmi les additionneurs les plus communs nous pouvons trouver le RCA (Ripple Carry Adder). Le RCA est un circuit logique dans lequel la retenue sortante (Cout) de chaque additionneur complet (FA) est connectée à la retenue entrante du FA de rang immédiatement supérieur. Les bits S et Cout de chaque FA ne peuvent être calculés qu'après l'arrivée de la retenue entrante correspondante. Autrement dit, afin d'obtenir le résultat de l'addition de deux nombres binaires utilisant un RCA, il faut attendre la propagation de la retenue à travers tous les étages de FA. Le temps de propagation des retenues à l'intérieur du circuit logique est déterminant pour la vitesse limitée au RCA. Le retard de propagation du bit Carry est le temps écoulé entre l'application de

l'entrée Carry-in et l'apparition de la sortie Carry-out correspondante. Par conséquence, le résultat final du RCA sera valide seulement après les retards de propagation conjoints de tous les FAs à l'intérieurs.

2.3. Approximate computing

Des travaux antérieurs sur l'Approximate Computing ont mis l'accent sur le calcul approximatif, en dehors de la mémoire approximative. Ces études comprennent l'application de la résilience à l'erreur dans un environnement approximatif [9]. Ce type de travail a aussi été fait sur les propositions de jeu d'instructions extensions au sommet des normes ISA régulières et / ou des changements de code lors de la compilation ou de l'exécution [12]. Pour contrôler l'approximation, des cadres adaptés pour l'Approximate Computing [13] [14] et des propositions pour les différents modèles d'architecture ont été faites [15] [16]. Il existe également des études appliquées avec un objectif spécifique, pour voir comment certains opérateurs se comportent dans un environnement de l'Approximate Computing, par exemple un encodeur vidéo en action avec des injections d'erreurs simulées [17]. Tous les ouvrages mentionnés montrent des résultats différents mais positifs.

Microsoft Research a proposé une mémoire DRAM modifiée, où une partie de la mémoire conserve une certaine fréquence de rafraîchissement, tandis qu'une seconde partie est autorisée à avoir un taux plus faible pour économiser l'énergie [18]. L'idée est de laisser les données moins critiques dans la partie avec un taux de rafraîchissement plus faible. Cette idée est reprise en outre par Sampson et al, où le moyen explicite de décider quelles données peuvent être approximatives est pris en charge [21]. Et puis, ils proposent le SRAM avec une alimentation de puissance plus faible, qui économise de l'énergie mais peut entraîner une perte de qualité.

Pour connaître la viabilité de l'Approximate Computing, des études ont été effectuées afin d'identifier quel type de matériel peut servir de base à ce paradigme. En particulier, "phase changing memories" ont été étudiées en raison de leurs capacités pour conserver les données même lorsqu'il n'y a plus d'alimentation. De plus, un modèle plus conservateur de type écrire-et-vérifier a été proposé au prix d'une petite possibilité de perte de qualité. Les mémoires de changement de phase sont donc d'un grand intérêt pour la recherche [19] [20] [22].

Approximation dans matériaux s'agit d'un objet matériel qui a une certaine tolérance des erreurs provoquées par diverses sources, qui autrement a besoin de mécanismes de correction explicites aux niveaux plus élevés, par exemple, la détection de bits etc. Différentes technologies peuvent dégrader la qualité des données différemment et les

erreurs peuvent se produire à cause de nombreuses raisons. Quelques facteurs le plus communs sont listés ci-dessous : les fluctuations de température, changements dans matériaux au fil du temps, drift au fil du temps, perte de charge électrique, influence des rayons cosmiques, défaillance matérielle statique. L'importance de l'identification des caractéristiques du matériel a été un sujet pendant longtemps afin de garantir la précision lors de l'utilisation. Dans Approximate Computing, cela est encore important, mais plus d'un point de vue sur la façon donc certaines structures de récupération peuvent être assouplies en raison de la tolérance d'erreurs des données.

Approximate Computing a été intégré dans une variété de dispositifs et composants. Différentes technologies de mémoire, composants du processeur, et les unités de traitement offrent différents compromis. Ces applications peuvent être classées en fonction de leur cible d'optimisation, qui met en évidence que l'Approximate Computing permet aux concepteurs d'optimiser plusieurs métriques d'intérêts au prix d'une petite perte de QoR. Par exemple, il existe des mémoires SRAM approximatives, mémoires eDRAM et DRAM approximatives, mémoires non volatiles approximatives, techniques approximatives pour les GPUs et FPGAs et nombreux composants de processeurs. La mise en œuvre des différentes techniques proposées pour l'Approximate Computing a obtenue de grands succès dans plusieurs domaines. J'ai listé quelques domaines ci-dessous : traitement des images ou de multimédia [32] [33], traitement du signal [34] [35], machine Learning [36] [37], computation scientifique [38] [39], analyse financière [40] [41], recherche de base données [42] [43]

2.4. Stratégies pour l'approximation

Une fois que les variables et les opérations qui doivent être approximatives ont été identifiées, elles peuvent devenir approximatives à l'aide d'une variété de stratégies.

2.4.1. Utilisation de précision mise à l'échelle

Certains travaux ont été faits en changeant la précision (nombre de bits) des opérandes d'entrée ou intermédiaires pour réduire les besoins de stockage et de computation. Par exemple, Yeh et al. [23] proposent l'échelle dynamique de précision pour améliorer l'efficacité de l'animation basée sur la physique. Leur technique peut trouver la précision minimale requise en effectuant le profilage au cours de design.

2.4.2. Utilisation de la boucle elliptique

Il existe également les techniques utilisant une approche de boucle elliptique, qui fonctionne par le saut de quelques itérations d'"une boucle pour réduire le frais généraux de computation. Sidiroglou et al. [24] identifient plusieurs modèles de calcul génériques qui fonctionnent bien avec boucle elliptique, telles que la simulation de Monte Carlo, le raffinement itératif, et l'énumération de recherche d'espace. Par exemple, dans l'énumération de recherche d'espace, la computation elliptique saute certains éléments et retourne l'un des éléments restants dans l'espace de recherche.

2.4.3. Utilisation de la charge de valeur approximative

Sur une miss de charge dans un cache, les données doivent être extraites de la mémoire cache de niveau suivant ou de la mémoire principale, ce qui entraîne une grande latence. LVA (Load Value Approximation) tire parti de la nature approximative des applications pour estimer les valeurs de charge, permettant ainsi à un processeur de progresser au lieu d'attendre pour une réponse. Miguel et al. [25] présentent une technique de LVA pour les applications graphiques. Par rapport aux indicateurs de charge de valeur traditionnelle, dans lesquels un bloc doit être récupéré sur chaque défaut de cache pour confirmer l'exactitude de la prédiction, leur technique récupère les blocs de temps en temps juste pour entraîner l'approximation. Par conséquence, la récupération d'un bloc de cache sur chaque défaut de cache n'est pas nécessaire, ce qui réduit l'accès à la mémoire de manière significative.

2.4.4. Utilisation de Mémorisation

Cette approche fonctionne en stockant les résultats des fonctions pour les réutilisations ultérieures avec des fonctions / entrées identiques. Grâce à la réutilisation de certains résultats, la capacité effective de mémorisation peut être agrandie au prix d'une approximation possible. Cette approche est utilisée par plusieurs applications de Approximate Computing. Rahimi et al . [26] démontrent que l'architecture SIMD expose la localité des valeurs d'un programme parallèle à toutes ses voies. Sur cette base, ils proposent une technique qui réutilise le résultat d'une instruction à travers différentes voies parallèles de l'architecture SIMD pour réduire leur coût global d'erreur de temps de récupération.

2.4.5. Utilisation de plusieurs versions du programme inexact

Nous discutons maintenant les techniques qui utilisent plusieurs versions de code d'application avec différents compromis entre la précision et les coûts globaux. Samadi et al . [27] présentent un ACT (Approximate Computing Technique) pour les

GPUs qui permettent de faire des compromis entre la performance et la précision en fonction de la métrique spécifiée par l'utilisateur. Leur technique fonctionne en deux phases. Dans la phase de compilation en mode hors connexion, un compilateur statique est utilisé pour créer plusieurs versions de noyaux CUDA avec différents niveaux de précision. Dans la phase de gestion du noyau d'exécution, un certain algorithme (a greedy algorithm) est utilisé pour ajuster les paramètres de noyaux approximatifs pour trouver les configurations les mieux adaptées pour atteindre l'objectif de la qualité souhaitée .

2.4.6. Utilisation du matériel inexact ou défectueux

Ganapathy et al . [28] présentent une technique pour réduire l'amplitude minimale des erreurs lors de l'utilisation des mémoires non fiables, ce qui est en contraste avec la technique ECC qui corrige les erreurs réellement. Sur chaque écriture, le mot de données est décalé circulairement pour mémoriser les bits les moins significatifs dans les cellules défectueuses de la mémoire. Cela pousse les erreurs vers les bits de poids plus faibles, ce qui réduit en conséquence l' amplitude d'erreurs de la sortie. Pour trouver un équilibre entre la qualité et la performance / énergie / région, leur technique permet une adaptation de la granularité du brassage de bits. Ils montrent que leur technique permet d'obtenir une amélioration significative avec la latence, la puissance et la zone par rapport à l'utilisation d' ECC. Yetim et al . [29] étudient la protection des erreurs minimum imposée à une micro-architecture pour atténuer le contrôle, l'adressage mémoire et les défauts d'accès E/S afin de permettre l'exécution approximative sur un processeur défectueux. Ils utilisent un MIS (Marco Instruction Sequencer), un MFU (Memory Fence Unit) et une approche de I/O, qui tous cherchent à contraindre l'exécution en fonction des informations de profilage de l'application. Ils montrent que même avec des erreurs assez fréquentes dans les applications vidéo et audio, leur technique fournit toujours une bonne qualité de sortie et évite les accidents et suspensions.

2.4.7. Utilisation de la tension mise à l'échelle

La mise à l'échelle de la tension d'alimentation réduit la consommation d'énergie des circuits au coût des erreurs possibles[30]. Par exemple, la réduction de la tension d'alimentation SRAM permet de diminuer le courant de fuite, mais va augmenter la possibilité d'une lecture erronée et d'un échec d'écriture. Certaine application a été faites en utilisant la mise à l'échelle de tension tout en tenant en compte des

compromis. Chippa et al . [31] présentent une application qui utilise l'approximation à plusieurs niveaux d'abstraction. Par exemple, au niveau du circuit une trop forte mise à l'échelle de tension est utilisée, sans la mise à l'échelle de la fréquence d'horloge. Le circuit d'addition est segmenté en additionneurs de plus petite largeur de bits. Sur la base de la mise à l'échelle de tension, la propagation de signal Carry à travers les points de segmentation est sous un contrôle adaptatif et les erreurs dues aux valeurs ignorées de Carry sont réduites en utilisant un circuit de correction à faible coût. Ils montrent que leur approche fournit une économie d'énergie importante avec une perte QoR (Quality of Result) mineure. Ils montrent également que l'utilisation d'un rapprochement entre niveaux offre beaucoup plus d'amélioration que l'utilisation de l'approximation à un seul niveau.

3. Additionneurs arithmétiques

Comme mentionné précédemment, les mises en œuvre d'un circuit approximatif ont été considérées comme une solution potentielle dans le but de réduire la consommation d'énergie. Sous ce contexte , de nombreux schémas d'approximation ont été proposés en réduisant le chemin critique et la complexité du matériel de l'additionneur précis pour répondre aux exigences de haute vitesse et de l'efficacité énergétique, ainsi que la nature de tolérance d'erreurs de certaines applications. Au cours de ma recherche des additionneurs approximatifs, j'ai beaucoup réfléchi sur les différences et des avantages/inconvénients des différents designs approximatifs par rapport à l'additionneur conventionnel. Afin de mieux comparer les performances des additionneurs approximatifs dans tous les sens, j'ai eu besoin d'obtenir mes propres résultats de simulations. Par conséquent, j'ai également écrit des codes Verilog pour chaque design avec lesquels j'ai pu faire des synthèses sous Cadence.

3.1. Quelques additionneurs approximatifs

3.1.1. LOA (Lower-Part-OR Adder)

Le LOA proposé dans [44] divise une addition de n bits en deux parties plus petites: la partie de (n- m) bits et l'autre de m bits . Comme cela est représenté dans la Figure.2.4, un LOA de n bits exploite un additionneur précis normal avec une largeur plus petite (que l'on appelle le sous-additionneur) qui calcule les valeurs précises du résultat des (n-m) bits les plus significatifs (MSB), ainsi que des portes OU qui se rapprochent du résultat des m bits les moins significatifs (LSB) en appliquant

l'opération binaire OU aux bits d'entrée respectifs. Une porte supplémentaire est utilisée pour générer un signal Carry à '1' pour le sous-additionneur de la partie supérieure lorsque les deux bits les plus significatifs dans la partie inférieure sont à '1'. Cela permet de prendre en compte le passage du Carry entre les deux parties de LOA pour diminuer son imprécision.

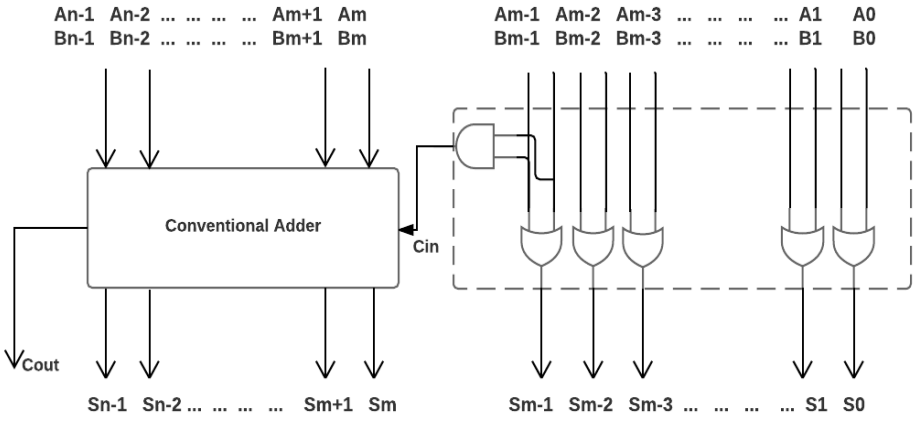


Figure 2.4 Schématique du LOA (Low-Part-OR Adder)

Comme il n'y a pas de chemin de propagation de retenue dans la partie inférieure des m bits, le délai du chemin critique peut être beaucoup plus petit que celui d'un RCA de n bits. Le retard total est constitué de deux parties: la première partie est le retard d'un (n- m) bits RCA, toutes les portes ou avec une unité ET composent l' autre partie du retard. Ce retard total est évidemment inférieur à celui d'un additionneur n bits.

Par rapport aux FAs utilisés comme unités les plus courantes dans additionneurs traditionnels, les portes OU utilisées dans les LSBs ont plus petite complexité du matériel, ce qui conduit à une efficacité en termes de surface. Avec la même idée, une brève étude analytique montre qu'il consomme moins d'énergie par rapport à un additionneur précis. La consommation dynamique d'énergie d'un circuit qui domine sa dissipation totale est donnée :

$$t_{LOA}=[t_{FA}+(N-M-1)t_{CP}]+[Max\{t_{MUX}, t_{OR}\}]$$

où a est le facteur d'activité, CL est la capacité de charge, VDD est la tension d'alimentation, et f est la fréquence d'horloge. Par conséquent, comme le LOA a moins de portes comme indiqué précédemment, il a une plus petite valeur de CL totale par rapport à un additionneur précis de la même longueur de mot. Dans le même temps, à cause de la partie inférieure qui est construite en utilisant des portes OU, il a aussi un facteur d'activité plus petit.

3.1.2. *SCSA (Speculative Carry Select Adder)*

Le SCSA [45] et les additionneurs avec latence variable sont conçus pour des entrées aléatoires non signées. Un SCSA de n bits est segmenté en (n / k) sous-additionneurs de k bits (fenêtre additionneurs) également. Comme ce qui est présenté dans la Figure 2.5, chaque sous-additionneur est constitué des deux petits additionneurs (adder0 et adder1) en utilisant n'importe quel additionneur classique. Adder0 calcule une somme et retenue avec un signal Carry-in à '0' pendant que adder1 calcule le résultat avec un Carry-in à '1'. Ensuite, les deux sommes possibles sont délivrées au multiplexeur et le Carry-out de sortie du adder0 dans l'ancien sous-additionneur est utilisé comme le signal de sélection.

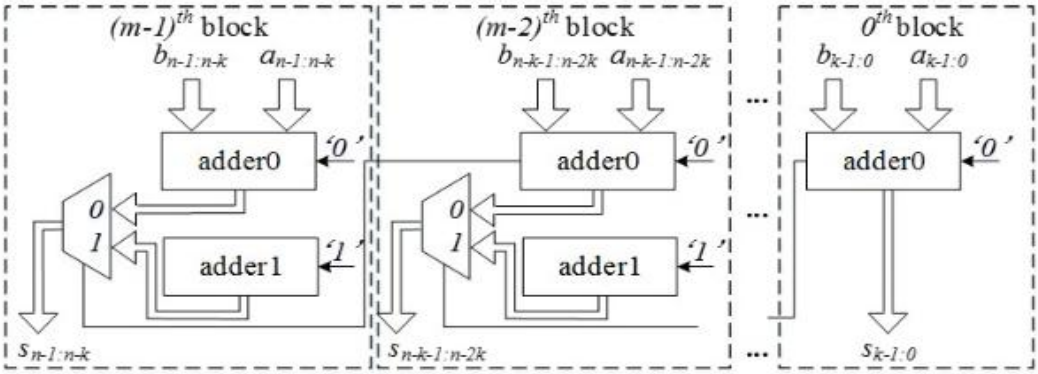


Figure 2.5 Schématique du SCSA (Speculative Carry Selection Adder)

Supposons que nous mettons en œuvre le petit additionneur dans l'additionneur de fenêtre en utilisant Kogge-Stone. Par conséquent, le retard du chemin critique d'un SCSA est O(log k), ce qui peut être beaucoup plus faible qu'un additionneur traditionnel (O(logn)).

En outre, un VLCSA (Variable Latency Carry Selection Adder) est proposé ensuite en ajoutant une détection et récupération d'erreur au SCSA. La détection d'erreur indique si la spéculation est incorrecte. La récupération d'erreur a produit un résultat correct lorsque la détection d'erreur signale la présence d'une erreur. Sur la base du modèle d'erreur analytique pour SCSA, le signal de détection d'erreur est défini comme suit:

$$\mathrm{ERR} = \sum_{i=0}^{\lceil \frac{n}{k} \rceil - 2} P_{k-1:0}^{i+1} G_{k-1:0}^{i}.$$

Ensuite, la partie de récupération d'erreur utilise les résultats intermédiaires de l'additionneur spéculatif pour calculer le bit Carry-out précis pour toutes les fenêtres.

Pour être plus précis, un additionneur préfix à $\lceil n / k \rceil$ bits prend les groupes signaux P/G de fenêtres comme entrées. Ainsi, longueur du chemin critique de récupération d'erreur est $O(\log k + \log \lceil n / k \rceil)$ .Mais pour les entrées gaussiennes signées, une partie non négligeable des Carry chaînes est aussi longue que la longueur de l'additionneur. Ces chaînes augmentent de manière significative la possibilité d'erreur, ainsi le VLCSA peut être plus lente que l'additionneur traditionnel.

3.1.3. ETAII (Error-Tolerant Adder II) et ETAIIM (ETAII Modifié)

Comme une amélioration de la conception antérieure ETA [46], l' ETAII proposé dans [47] peut résoudre le problème du calcul des entrées de petites nombres avec l'ancien design. ETAII ne supprime pas la totalité ou une partie du chemin de propagation de retenue comme l'ETA. Au contraire, il divise la totalité du chemin de propagation de retenue en un certain nombre de chemins courts, et il complète les propagations de retenue dans ces courts chemins simultanément. De cette façon, la performance de la vitesse de l'additionneur peut être améliorée fortement avec presque aucune dégradation de sa consommation d'énergie.

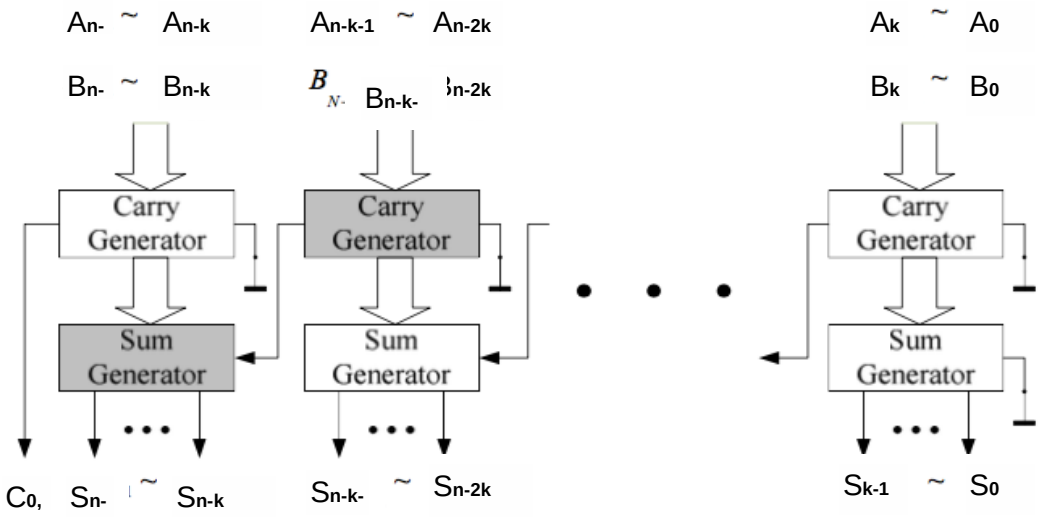


Figure 2.6 Schématique du ETAII

Un ETAII n bits est divisé en n / k blocs. Chaque bloc contient k bits et est constitué de deux circuits séparés (générateur de Carry et générateur de Somme) comme indiqué dans la Figure 2.6. Le générateur de Somme prend le signal Carry-in du bloc précédent alors que le générateur de Carry ne prend pas de Carry-in. Par conséquent, l'ETAII utilise plus d'information pour prédire le bit Carry et il est donc plus précis

par rapport à l'ETA avec le même k. Le chemin de propagation pire cas de cet additionneur est coloré en gris dans la figure 2.6.

L'ETAII est capable de parvenir à une amélioration de plus de 30 % avec le produit Power-Delay (PDP) par rapport à la conception ETA. Mais la précision dégradée pour les grands opérandes d'entrée peut restreindre son utilisation. Une structure modifiée qui est nommée ETAIIM est donc mise en œuvre pour améliorer encore les performances de précision de ETAII. Dans cette structure, les trois premiers générateurs de Carry sont montés en cascade pour générer les signaux Carry pour les deux blocs des rangs plus élevés. De cette façon, plus de bits d'entrée sont pris en compte lors du calcul du signal Carry pour le MSBs.

Ces deux modèles ont un chemin critique plus court par rapport au RCA. Avec environ 53% de surcharge de matériel, le PDP de l'ETAII peut être réduit de 78 % et pour l'ETAIIM la réduction de PDP est de 63%. Comme j'ai mentionné plus tôt, l'ETAII a des problèmes lors du calcul des grandes entrées. Pour les deux modèles, lorsque le pire cas arrive, une grande quantité de temps et d'énergie sera consommée le long du chemin de propagation de retenue. L'auteur ne propose pas de solution pour faire face à ce problème et se contente de dire que le pire cas se produit rarement.

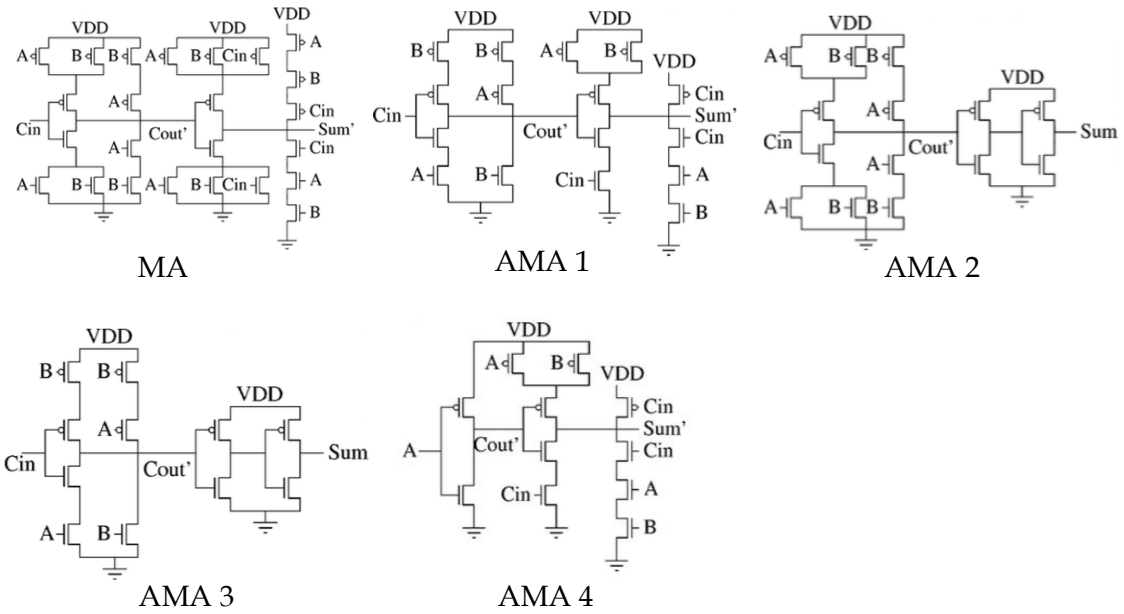


Figure 2.7 MA conventionnel et ses 4 versions approximatives

3.1.4. AMA (Approximate Mirror Adder)

Cinq additionneurs approximatifs AMAs sont proposés dans [48] en supprimant plusieurs transistors du circuit de l'additionneur classique MA (Mirror Adder) en

respectant certains critères. Aucune combinaison d'entrée devrait parvenir aux circuits cours/ouverts dans un nouveau schéma simplifié. Un autre critère est d'introduire un minimum d'erreurs dans table de vérité de l'additionneur complet (FA) pendant qu'on simplifie la conception classique. Ensuite, les cellules de FAs approximatives sont utilisées pour composer les additionneurs multi-bits où ces cellules approximatives ne sont utilisées que dans les LSBs, assurant ainsi que la qualité de la production finale ne se dégrade pas trop.

Inputs			Accurate Outputs		AMA1 Outputs		AMA2 Outputs		AMA3 Outputs		AMA4 Outputs	
A	**B**	**Cin**	**Cout**	**S**	**Cout**	**S**	**Cout**	**S**	**Cout**	**S**	**Cout**	**S**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**
0	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**
0	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**0**
0	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**1**
1	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**
1	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**
1	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**

Table 2.1 Table de vérité des AMAs

La stratégie d'approximation pour la MA peut être divisée en plusieurs petites étapes. L'idée principale est de supprimer certains transistors du MA classique en introduisant des erreurs à la table de vérité originelle. Par conséquent, les quatre AMAs seront introduits en illustrant ses tables de vérité, expressions simplifiées de Set Cout, et leurs conceptions au niveau de transistor sont présentés dans la Figure 2.7.

Ces réductions de complexité conduisent à une réduction de puissance de deux façons différentes. Tout d'abord, une réduction inhérente de la capacité et des résultats inexacts aident à avoir une plus petite taille du matériel. Deuxièmement, la réduction de la complexité conduit fréquemment aux chemins critiques plus courts, ce qui facilite la réduction de la tension sans erreur induite par synchronisation. Ces additionneurs approximatifs peuvent être utilisés dans le domaine du traitement d'image. La DCT (Discrete Cosine Transform) et l'IDCT (Inverse Discrete Cosine Transform) sont des composants intégrantes d'un système de compresion d'image

JPEG (Joint Photographic Experts Group) [49], et les deux blocs DCT et IDCT peuvent fonctionner à une tension d'alimentation inférieure grâce à l'utilisation de celles additionneurs approximatives mentionnées ci-dessus. Les résultats montrent que l'AXA5 fournit des économies d'énergie maximales entre tous les versions d'approximation. Il est intéressant de noter qu'il fournit environ 60% d'économie d'énergie lorsque 9 LSBs sont approximatifs, avec un PSNR de 25.46 dB. Dans le même scénario, la troncature peut fournir 61% d'économie d'énergie, mais la qualité de sortie est dégradée très significativement avec un PSNR de 13.87 dB.

3.1.5. AXA (Approximate XOR/XNOR-based Adder)

Inputs			Accurate Outputs		AXA1 Outputs		AXA2 Outputs		AXA3 Outputs	
A	**B**	**Cin**	**Cout**	**S**	**Cout**	**S**	**Cout**	**S**	**Cout**	**S**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**
0	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**
0	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**0**
1	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**
1	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**0**

Table 2.2 Table de vérité des AXAs

Les additionneurs AXA (Approximate XOR/XNOR-based Adder) proposés sont conçus basant sur les portes XOR/XNOR avec multiplexeurs mis en œuvre avec des transistors. AXA1 est basé sur l'additionneur complet [50] contenant 10 transistor, tandis que AXA2 et AXA3 sont basés sur le design précis proposé dans [51]. Les auteurs ont fait une évaluation dans [50] et comparés par leurs performances dans la consommation d'énergie , le retard , la superficie et le produit de retard et puissance (PDP) avec un additionneur complet précis (FA). Dans l'AXA1, l'opération XOR est mise en oeuvre par un inverseur et deux transistors connectés au X et Y respectivement. Le nombre de transistors utilisés dans cette conception est 8. La conception de AXA2 met en oeuvre un additionneur approximatif avec six transistors : une porte XNOR de 4 transistors et un bloc de transistors. Et l'AXA3 est une extension d'AXA2 qui utilise 2 transistors de plus dans une configuration de transistors pour une meilleure précision de Somme. Les fonctions de somme (S) et

Cout sont ensuite obtenues à partir de la table de vérité (Table 2.4) ci-dessus et les trois modèles approximatifs sont présentés dans la figure 2.8.

AXA1 : quatre erreurs dans Sum (S), quatre erreurs dans Cout.

$$S = C_{in} \quad ; \quad C_{out} = \overline{(A \oplus B)C_{in} + \overline{A}\,\overline{B}} \quad ;$$

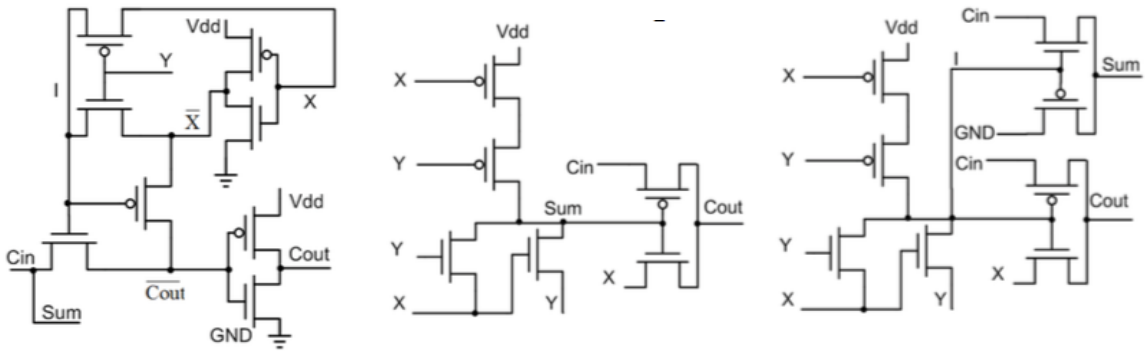
AXA2: quatre erreurs dans Sum (S), Cout exact.

$$S = \overline{(A \oplus B)} \quad ; \quad C_{out} = (A \oplus B)C_{in} + AB \quad ;$$

AXA3: deux erreurs dans Sum (S), Cout exact.

$$S = \overline{(A \oplus B)}\,C_{in} \quad ; \quad C_{out} = (A \oplus B)C_{in} + AB \quad ;$$

La comparaison avec l'additionneur complet précis (FA) montre qu'avec une perte très faible de précision, des économies évidentes dans le nombre de transistors et la puissance peuvent être obtenues. L'évaluation de ces designs montre également un meilleur temps de propagation qu'un additionneur précis (sauf pour le retard de Carry-out de AXA2) et par conséquent ces additionneurs sont plus rapides en cascade pour une addition des mots longues. Les auteurs tirent les conclusions que l'AXA1 a la meilleure performance (par exemple, un délai le plus court), l'AXA2 utilise la plus petite surface et l'AXA3 est la conception la plus économe en énergie avec la distance d'erreur la plus courte. Dans l'ensemble, AXA1 a le plus faible PDP.



AXA1 AXA2 AXA3

Figure 2.8 Les 4 versions approximatives de AXA

3.2. Conception du code Verilog et synthèses sous Cadence

J'ai voulu établir une comparaison des additionneurs approximatifs dans tous les sens en prenant en compte le plus grand nombre des additionneurs possible. De plus, considérant que les conclusions des performances d'un certain nombre

d'additionneurs sont tellement variées entre différents auteurs, je me suis obligée de réaliser des simulations sous Cadence pour obtenir mes propres résultats.

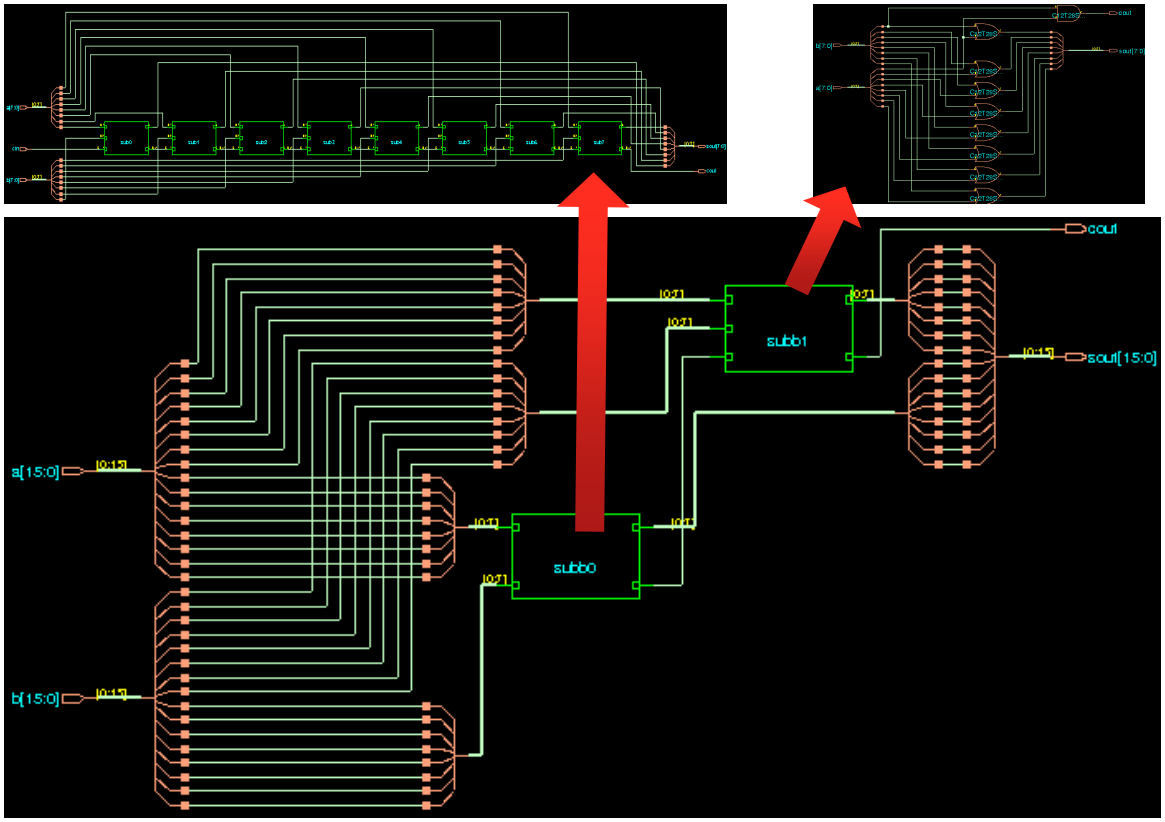


Figure 2.9 Circuit du LOA après synthèses

Pour ce qui concerne les synthèses sous Cadence, il existe plusieurs chapitres qui traitent des instructions dans le tutoriel officiel. C'est vraiment important de configurer les chemins et les bibliothèques correctement utilisant des bonnes instructions, qui a été un grand obstacle pour moi avant continuer avec mes synthèses. Un exemple des circuits conçus après la synthèse est présenté dans la Figure 2.9. Comme ce qui est présenté dans la Table 2.3, une comparaison des performances en termes du retard, la consommation d'énergie, la taille matérielle et le PDP est établie entre nombreux additionneurs approximatifs.

Le chemin critique le plus long pour l'additionneur classique présenté dans la figure 2.10 (a) apparaît lorsque les signaux propagateurs dans toutes les positions binaires sont égales à '1', ce qui conduit à Cn -1 = Cin. Ainsi, afin de réduire le retard de trajet critique, une méthode efficace consiste à réduire la longueur de la chaîne de propagation de retenue.

Name	cells	Cell area	Net area	Total area	leakage (nW)	Internal (nW)	Net (nW)	Switching (nW)	Worst Delay	PDP (10^-18)
RCA (8 bits)	8	19.75	26.96	46.71	13.05	1229.16	6008.55	7237.71	534.00	3864.56
RCA (16 bits)	16	40.64	53.92	94.56	26.74	2649.57	12481,80	15131.37	1038.00	15705.98
RCA (32 bits)	32	82.42	107.8	190.26	54.12	5656.29	342.14	31371.46	2044.40	63997.78
app ETA (16)	40	42.11	83.82	125.93	26.75	1601.00	8198,30	9799,30	583,10	5713.80
ACAA(16/8)	24	48.14	65.54	113.69	29.29	2737.74	12621.66	15359.40	530.40	8146.41
ACAA (16/4)	28	49.45	68.83	118.28	27.70	2514.31	12144.73	14659.04	279.60	4098.66
ACAA(32/8)	56	104.9	142.7	247.65	61.75	6122.36	26661.02	32783.38	531.20	17414.33
SCSA(16/4)	52	85.68	105.3	191.06	56.64	5015.29	14915.60	19930.90	356.40	7103.05
LOA(16)	17	26.76	48.84	75.61	18.16	1374.18	7668.42	9042.60	465.10	4205.43
LOA(32)	33	52.88	96.03	148.90	35.85	3185.30	16621.24	19806.54	1096.80	21700.80
LOA(8+24)	33	37.21	89.29	126.49	26.17	1828.13	11556.84	13384.97	593.60	7945.34
LOA(24+8)	17	26.76	48.84	76.61	18.16	1359.64	7618.22	8977.86	593.60	5328.75
AMA1(16bits)	45	39.17	82.50	121.67	28.52	1611.02	7407.70	9018.72	775.70	6995.26
AMA2(16bits)	62	32.64	85.76	118.40	21.47	1105.70	8872.66	9978.37	2654.60	26487.60
AMA3(16bits)	46	27.42	84.60	112.02	19.44	690.07	5862.80	6552.86	814.50	5336.60
AMA4(16bits)	31	17.63	58.39	76.02	16.47	636.82	5018.62	5655.45	119.60	676.34
AMA5(16bits)	17	8.32	28.65	36.97	5.29	258.41	3413.81	3672.22	119.60	439.17
AMA1(32bits)	93	80.95	169.6	250.55	58.76	3426.43	15357.73	18784.16	1572.50	29537.84
AMA2(32bits)	126	66.59	173.2	239.80	43.86	2255.72	17594.72	19850.44	5137.00	101969.45
AMA3(32bits)	94	56.14	171.7	227.84	39.99	1533.18	11942.23	13475.41	1634.50	22024.89
AMA4(32bits)	63	35.90	118.8	154.79	33.70	1142.51	9054.07	10196.58	119.60	1219.44
AMA5(32bits)	33	16.16	55.61	71.76	10.27	497.82	6576.61	7074.43	119.60	846.05
new1(32/8)	94	107.7	184.2	291.94	78.99	5750.36	26577.91	32328.27	617.50	19962.54
new1(32/4)	94	107.7	185.5	293.24	78.96	5778.69	26926.59	32705.26	401.90	13144.14

Table 2.3 Comparaison des additionneurs approximatifs étudiés

L'idée principale est de calculer le signal générateur et le signal propagateur en premier lieu utilisant des opérandes. Et puis dans les étapes suivantes, ces deux signaux générés précédemment sont utilisés comme les entrées dans les prochains calculs. Une stratégie de spéculation de retenue est alors introduite dans le générateur de retenue (Carry Generator) : le signal générateur du bit le plus significatif dans le bloc précédent sert au signal Carry-in du bloc Carry Generator. Par conséquent, la

sortie du bloc i de Carry Generator est maintenant une approximation de Carry-out par rapport au résultat correct, qui est désigné.

Comme ce qui est représenté sur la Figure 2.10 (b), le Carry-out approximatif est utilisé comme signal Carry-in au générateur de somme (Sum Generator). Ainsi, les bits de somme au bloc i se composent également d'un résultat approximatif désigné. La longueur du chemin le plus long de propagation de retenue est ainsi réduite de n à 2k, ce qui réduit considérablement le retard total du circuit. Cependant, seulement un signal de Carry-out spéculé dans le bloc i, qui pourrait être erroné.

4. Les Résultats Obtenus

Selon ma recherche sur les additionneurs approximatifs, on peut travailler avec le design d'additionner 1bit ou avec la stratégie utilisée dans un multi-bit additionneur afin de réaliser l'approximation. Premièrement, nous pouvons introduire des erreurs dans la table de vérité de l'additionneur complet classique (FA). Il existe plusieurs façons d'arrangement des erreurs, et les erreurs peuvent apparaître dans la somme (S) et le retenue (Cout) en même temps. L'étape suivante est d'écrire les nouvelles représentations de S et Cout en fonction des opérandes d'entrée. Cela peut également nous donner une impression imprécise si le design peut être plus simple que l'ancien design. Et puis nous arrivons finalement à la conception du circuit utilisant des résultats obtenus dans les deux étapes précédentes. Le circuit peut être conçu utilisant des portes logiques ou en éliminant de certains transistors à partir des circuits conventionnels.

Pour ce qui concerne l'additionneur de multi-bit, l'idée essentielle est de réduire le chemin critique de propagation des signaux et/ou la complexité matérielle par rapport aux designs conventionnels. D'après mes études, j'ai catalogué les approches appliquées dans les conception des nouvelles additionneurs approximatifs en trois stratégies principales :

1) Utiliser des additionneurs approximatifs de bit unique dans une partie (ou plusieurs parties) du design conventionnel, normalement dans la partie des bits moins importants.

2) Diviser un additionneur en plusieurs sous-additionneurs plus petits, ainsi la longueur du chemin de propagation des signaux sera normalement beaucoup réduite.

3) Remplacer les vrais signaux Carry-in entre différentes unités de calcul par les Carry-in prédictifs. Par conséquent, le retard du chemin critique de propagation du

signal sera plus court puisqu'il n'a plus besoin d'attendre l'apparition de vrais Carry-in pour commencer le calcul suivant.

4.1 Analyse d'erreurs

La distance d'erreur (ED) et la distance moyenne d'erreur (MED) sont proposées dans [54] pour évaluer la performance arithmétique des circuits approximatives. Pour un additionneur approximative à n bits, ED est définie comme la valeur absolue de la différence entre les sommes approximatives et précises, i.e.,

$$ED = |S' - S|$$

où S' est la somme de l'additionneur approximatif et S est la somme d'un additionneur précis pour une combinaison d'entrée avec une précision donnée. La MED est définie comme la moyenne d'ED pour un ensemble de vecteurs d'entrée, i.e.,

$$MED = \mathrm{E}[ED] = \sum_{i=0}^{S_{MAX}} iP(i)$$

où SMAX est la puissance maximale d'un additionneur à n bits (i.e., 2n+1 - 1) et P(i) est la probabilité de ED étant à i. Le taux d'erreur (ER) est défini comme le pourcentage de résultats erronés parmi toutes les sorties [55], i.e.,

$$ER = \sum_{i=0}^{S_{MAX}} P(i)$$

Les mesures ci-dessus (ER et MED) sont intéressantes pour évaluer la performance arithmétique additionneurs approximatifs.

4.2 Synthèse et Conception du circuit électronique

En utilisant le kit de conception de 28nm UTBB-FDSOI à la base de la description en Verilog de conception du nouveau additionneur, nous avons pu réaliser la synthèse de l'additionneur approximatif proposé. Le circuit de cet additionneur est présenté dans la Figure 2.11.

De plus, la synthèse a permis d'effectuer aussi une simulation transitoire du circuit. Les performances de l'additionneur à 32 bits dans l'aspect du retard, la consommation d'énergie, la taille matérielle et le PDP sont présentées dans la Table 2.4. Plus précisément, les deux dernières lignes correspondent aux deux stratégies différentes utilisant des sous-additionneurs à 8 ou 4 bits (Table 2.2).

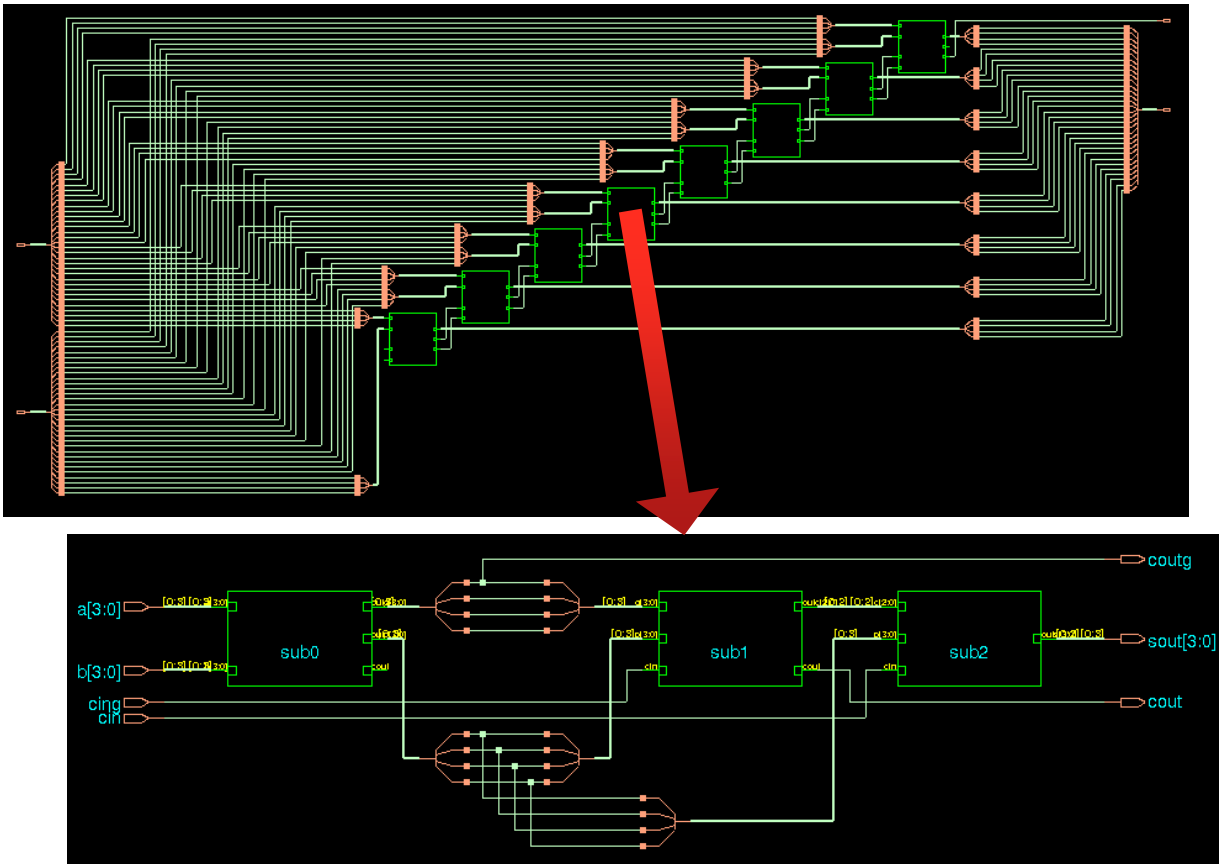


Figure 2.11 Synthèse du nouveau additionneur sous-additionneurs de type 4 bits

Name	cells	Cell area	Net area	Total area	leakage (nW)	Internal (nW)	Net (nW)	Switching (nW)	Worst Delay (ps)	PDP (10^-18)
new1(32/8)	94	107.71	184.23	291.94	78.99	5750.36	26577.91	32328.27	617.50	19962.54
new1(32/4)	94	107.71	185.53	293.24	78.96	5778.69	26926.59	32705.26	401.90	13144.14

Table 2.4 Performances du nouveau additionneur approximatif

La conception du circuits a été effectuée sous Cadence utilisant des transistors et des éléments basiques. Les assemblages de transistors qui correspondent aux additionneurs de 4, 8 et 16 bits ont donné la possibilité de réaliser des simulations. Considérant que cette conception représente des opérations séquentielles, la présence des signaux de contrôle et la gestion de l'horloge sont importantes dans ce circuit.

Dans Figure 2.12, le circuit d'un additionneur à 8 bits est présenté, et puis dans Figure 2.14 vous pouvez trouver son résultat de simulation pendant une période de 300ns.

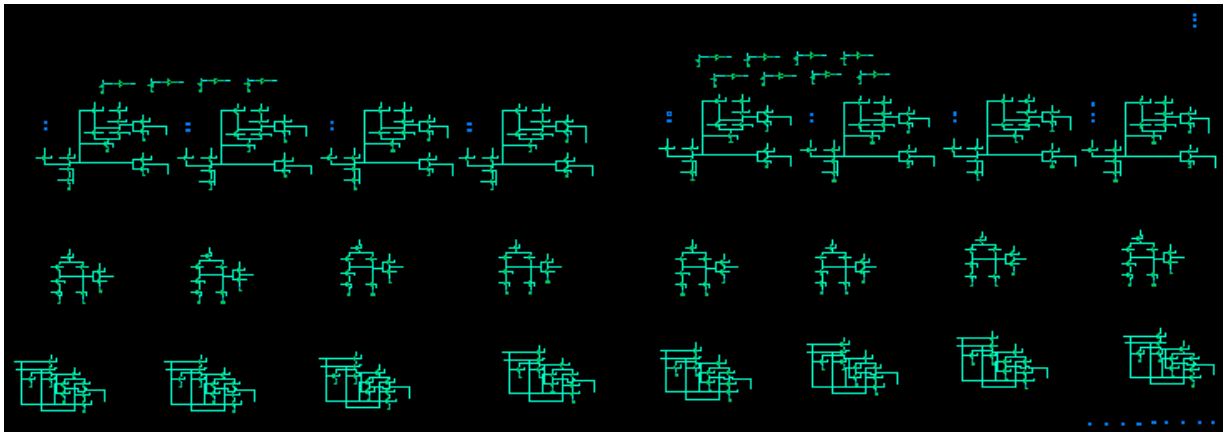


Figure 2.12 Conception du circuit électronique sous Cadence de l'additionneur

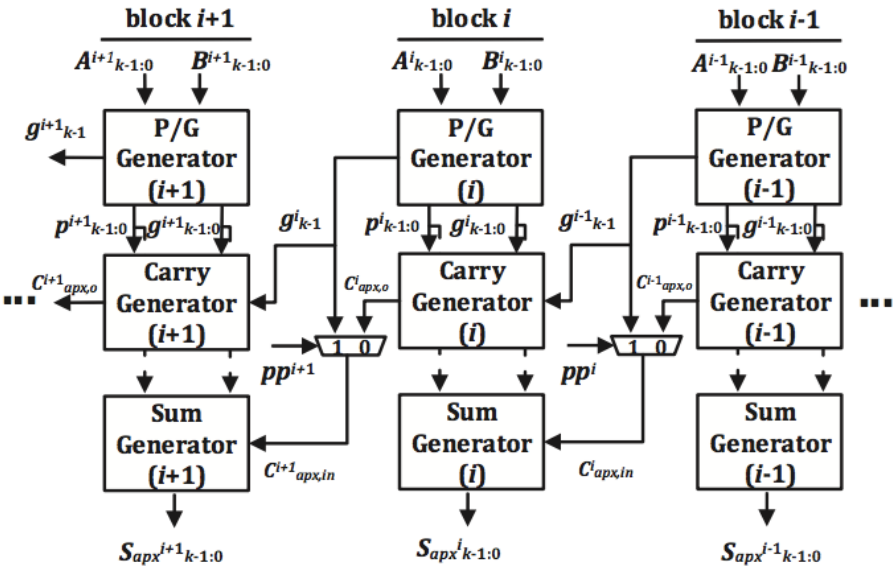


Figure 2.13 Version modifiée de l'additionneur approximatif

Dans le nouveau additionneur approximatif présenté, c'est possible que la somme approximative commence par 00 ... 0, tandis que le somme correcte devrait commencer par 10. . . 0. Cela pourrait conduire à une erreur relative jusqu'à 100%. De même façon, grandes erreurs relatives peuvent être obtenues pour les cas où $pp^i = 1$, $g^{i-1}\boldsymbol{k\text{-}1} = 0$, $C^{i-1}_{apx,0} = 1$ et tous les bits d'entrée pour les blocs à la gauche du bloc i sont à '0'.

Pour réduire la grande erreur relative, l'auteur a présenté dans [53] un autre module avec une réduction d'erreur relative par rapport à l'ancien additionneur. L'additionneur approximatif modifié est indiqué dans Figure 2.13, où il insère seulement un multiplexeur 2 à 1 entre chaque paire de blocs adjacents. Les deux entrées du multiplexeur sont les signaux générateurs du bit le plus significatif dans le bloc

précédent, q^{i-1}***k-1*** , et le signal Carry-out produit par le générateur de Carry dans le bloc précédent $C^{i-1}_{apx,0}$. Le signal de sélection a été généré dans le générateur de Carry du bloc i, il n'y a pas besoin d'inclure des circuits additionnels pour l'obtenir. Le signal de sortie du multiplexeur est utilisé comme signal Carry-in du générateur de somme du bloc courant, qui est désigné par C^{i}_{in}. Cette amélioration provient de l'idée d'abaisser la position de bit où l'erreur se produit, ce qui est obtenu en forçant certains bits dans des positions inférieures à être faux plutôt que de laisser qu'un bit dans la position supérieure soit incorrect. Cette modification n'engendre presque aucun surcoût matériel à l'ancien design approximatif.

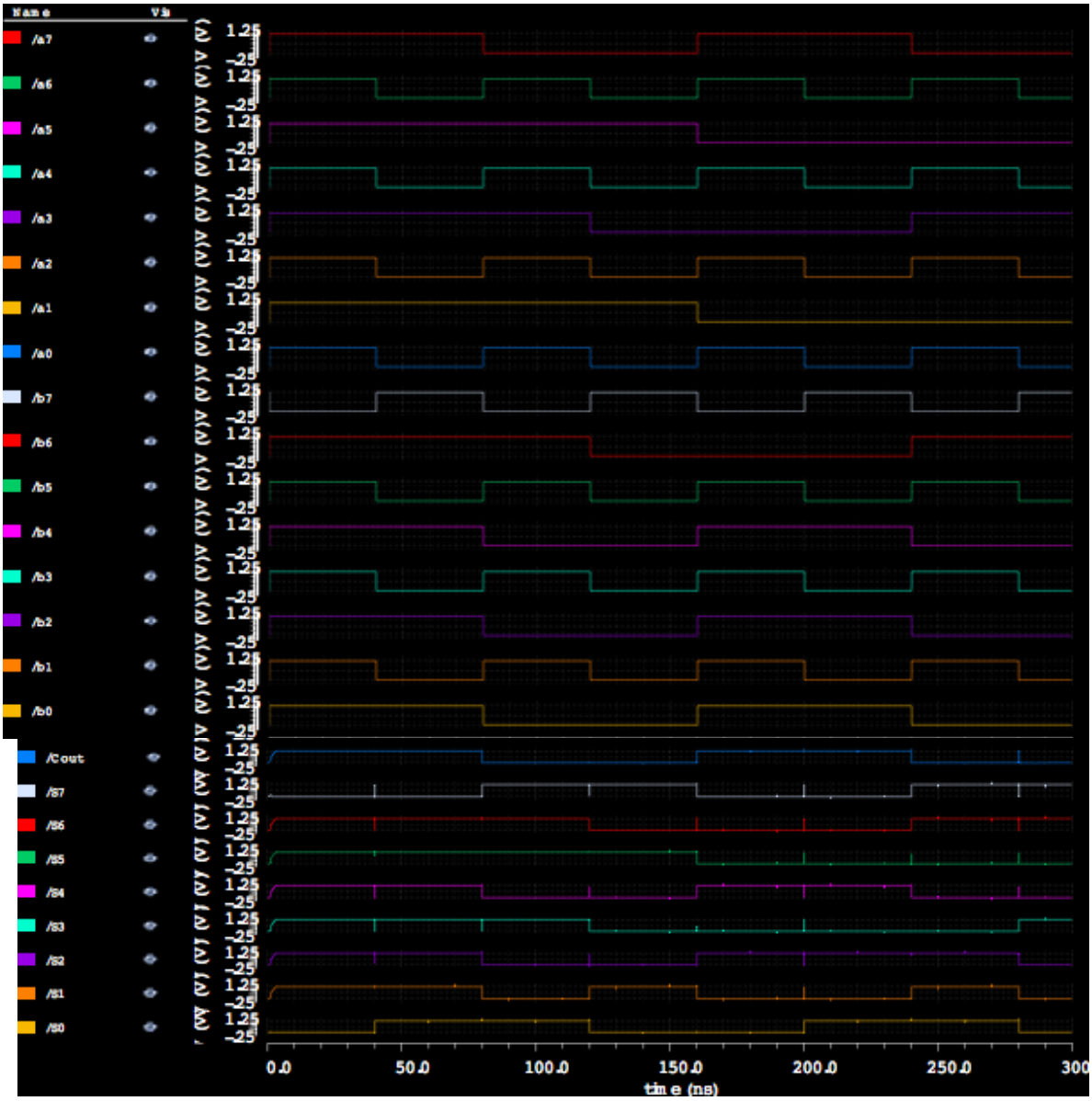


Figure 2.14 Résultat de simulation de l'additionneur présenté à 8 bits

Références Bibliographiques

[1] L. Naviner, H. Cai, Y. Wang, W. Zhao, Stochastic Computation With Spin Torque Transfer Magnetic Tunnel Junction, The 13th IEEE International NEW Circuits And Systems (NEWCAS) conference, Grenoble, France, 2015

[2] H. Cai, Y. Wang, W. Zhao, L. Naviner: Multiplexing Sense Amplifier Based Magnetic Flip Flop in 28nm FDSOI Technology, IEEE Trans. on Nanotechnology, Vol 14, issue 4, p. 761-767, 2015.

[3] Y. Wang, H. Cai, L. Naviner ; Y. Zhang ; X. Zhao ; E. Deng ; J. O. Klein ; W. Zhao: Compact Model of Dielectric Breakdown in Spin Transfer Torque Magnetic Tunnel Junction, IEEE Transaction on Electron Device, Vol 14, issue 4, p. 1762-1767, 2015

[4] H. Cai, Y. Wang, L. Naviner, W. Zhao, “Low Power Magnetic Flip-Flop Optimization With FDSOI Technology Boost", IEEE Transaction on Magnetic, Vol 52, issue 8, 2016

[5] L. Li and H. Zhou. “On error modeling and analysis of approximate adders”, In ICCAD, pages 511–518, 2014.

[6] H. Cai, Y. Wang, L. Naviner, W. Zhao: Breakdown Analysis of Magnetic Flip-flop With 28nm UTBB FDSOI Technology, IEEE Transaction on device and material reliability, 2016, Vol 16, issue 3, p. 376-383, 2016

[7] Y. Wang et al, “A novel circuit design of true random number generator using magnetic tunnel junction”, 12th ACM/IEEE International Symposium on Nanoscale Architectures (Nanoarch 2016), At Beijing, China

[8] International Roadmap for semiconductor (ITRS) 2011, ERD Update.

[9] Y. Wang, Y. Zhang, E. Deng, J. Klein, L. Naviner, and W. Zhao, “Compact model of magnetic tunnel junction with stochastic spin transfer torque switching for reliability analyses”, *Microelectronics Reliability*, vol. 54, no. 910, pp. 1774–1778, 2014.

[10] M. Marins de Castro et al., “Processional spin-transfer switching in a magnetic tunnel junction with a synthetic anti-ferromagnetic perpen-dicular polarizer”, *J Appl. Phys.*, 2012:111:07C912.

[11] S. Matsunaga, J. Hayakawa, S. Ikeda, K. Miura, T. Endoh, H. Ohno, and T. Hanyu, “MTJ-based nonvolatile logic-in-memory circuit, future prospects and issues”, in *Design, Automation Test in Europe Conference Exhibition, 2009. DATE ’09.*, April 2009, pp. 433–435.

[12] S. Sidiroglou-Douskos, S. Misailovic, H. Hoffmann, and M. Rinard, "Managing performance vs. accuracy trade-offs with loop perforation", in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pp. 124–134, ACM, 2011.

[13] M. De Kruijf, S. Nomura, and K. Sankaralingam, "Relax: An architectural framework for software recovery of hardware faults", in *ACM SIGARCH Computer Architecture News*, vol. 38, pp. 497–508, ACM, 2010.

[14] W. Baek and T. M. Chilimbi, "Green: a framework for supporting energy-conscious programming using controlled approximation", in *ACM Sigplan Notices*, vol. 45, pp. 198–209, ACM, 2010.

[15] S. Narayanan, J. Sartori, R. Kumar, and D. L. Jones, "Scalable stochastic processors", *in Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 335–338, European Design and Automation Association, 2010.

[16] H. Cho, L. Leem, and S. Mitra, "Ersa: Error resilient system architecture for probabilistic applications", *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 31, no. 4, pp. 546–558, 2012.

[17] Y. Fang, H. Li, and X. Li, "A fault criticality evaluation framework of digital systems for error tolerant video applications", in *Test Symposium (ATS), 2011 20th Asian*, pp. 329–334, IEEE, 2011.

[18] S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn, "Flicker: Saving refresh-power in mobile devices through critical data partitioning", *tech. rep., Technical Report. Microsoft Research*, 2009.

[19] H. Cai, Y. Wang, W. Zhao, L. Naviner: Ultra wide voltage range consideration of reliability-aware STT magnetic flip-flop in 28nm FDSOI technology, Microelectronics Reliability, 2015, Vol 55, issue 9, p. 1323-1327, 2015

[20] Y. Wang, H. Cai, W. Zhao, L. Naviner: Compact thermal modeling of spin transfer torque magnetic tunnel junction, Microelectronics Reliability, 2015, Vol 55, issue 9, p. 1649-1653. 2015

[22] H. Cai, Y. Wang, K. Liu, L. Naviner, H. Petit, J-F. Naviner, Cross-layer investigation of continuous-time sigma–delta modulator under aging effects, Microelectronics Reliability. Vol. 55, no. 3, p. 645-653, 2015

[23] Thomas Y. Yeh, Petros Faloutsos, Milos Ercegovac, Sanjay J. Patel, and Glenn Reinman, "The art of de- ception: Adaptive precision reduction for area efficient physics acceleration", In *International Symposium on Microarchitecture,* pp.394-406, 2007.

[24] Stelios Sidiroglou, Sasa Misailovic, Henry Hoffmann, and Martin Rinard, "Managing performance vs. accuracy trade-offs with loop perforation", *ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering,* pp.124–134, 2011.

[25] Joshua San Miguel, Mario Badr, and Enright Natalie Jerger, "Load value approximation", *MICRO,* 2014.

[26] Azar Rahimi, Luca Benini, and Rajesh K. Gupta, "Spatial memoization: Concurrent instruction reuse to correct timing errors in SIMD architectures", *IEEE Transactions on Circuits and Systems II: Express Briefs* 60, pp.847–851, 2013.

[27] Mehrzad Samadi, Janghaeng Lee, D. Anoushe Jamshidi, Amir Hormati, and Scott Mahlke, "SAGE: Self-tuning approximation for graphics engines", In *International Symposium on Microarchitecture*. pp.13– 24, 2013.

[28] Shrikanth Ganapathy, Georgios Karakonstantis, Adam Shmuel Teman, and Andreas Peter Burg, "Mitigating the impact of faults in unreliable memories for error-resilient applications", In *Design Automation Conference*, 2015.

[29] Yavuz Yetim, Margaret Martonosi, and Sharad Malik, "Extracting useful computation from error-prone processors for streaming applications", In *Design, Automation and Test in Europe Conference and Exhibition (DATE'13),* pp.202–207, 2013.

[30] Sparsh Mittal, "A survey of architectural techniques for near-threshold computing", *ACM Journal on* Emerging Technologies in Computing Systems 12, 4, pp.46:1–46:26, 2015.

[31] Vinay K. Chippa, Debabrata Mohapatra, Kaushik Roy, Srimat T. Chakradhar, and Anand Raghunathan, "Scalable effort hardware design", *IEEE Transactions on Very Large Scale Integration (VLSI) Sys- tems* 22, 9, 2014.

[32] Sasa Misailovic, Michael Carbin, Sara Achour, Zichao Qi, and Martin C. Rinard, "Chisel: Reliability- and accuracy-aware optimization of approximate computational kernels", In *International Conference on* Object Oriented Programming Systems Languages and Applications. 309–328, 2014.

[33] Woongki Baek and Trishul M. Chilimbi, "Green: A framework for supporting energy-conscious pro- gramming using controlled approximation", In *ACM SIGPLAN Notices*, Vol. 45. 198–209, 2010.

[34] Swagath Venkataramani, Amit Sabne, Vivek Kozhikkottu, Kaushik Roy, and Anand Raghunathan, "SALSA: Systematic logic synthesis of approximate circuits", In *Design Automation Conference*. 796–801,2012.

[35] Renée St. Amant, Amir Yazdanbakhsh, Jongse Park, Bradley Thwaites, Hadi Esmaeilzadeh, Arjang Hassibi, Luis Ceze, and Doug Burger, "General-purpose code acceleration with limited-precision analog computation", In *International Symposium on Computer Architecture*. 505–516, 2014.

[36] Jason Ansel, Yee Lok Wong, Cy Chan, Marek Olszewski, Alan Edelman, and Saman Amarasinghe, “Lan- guage and compiler support for auto-tuning variable-accuracy algorithms”, In *International Symposium on Code Generation and Optimization*. 85–96, 2011.

[37] Thierry Moreau, Mark Wyse, Jacob Nelson, Adrian Sampson, Hadi Esmaeilzadeh, Luis Ceze, and Mark Oskin, “SNNAP: Approximate computing on programmable SoCs via neural acceleration”, In *Inter*- national Symposium on High Performance Computer Architecture (HPCA’15). 603–614, 2015.

[38] Michael Carbin, Sasa Misailovic, and Martin C. Rinard, “Verifying quantitative reliability for programs that execute on unreliable hardware”, In *ACM SIGPLAN Notices*, Vol. 48. 33–52, 2013.

[39] Beayna Grigorian and Glenn Reinman. “Accelerating divergent applications on SIMD architectures using neural networks”, *ACM Transactions on Architecture and Code Optimization (TACO) 12*, 2015.

[40] Divya Mahajan, Amir Yazdanbakhsh, Jongse Park, Bradley Thwaites, and Hadi Esmaeilzadeh, “Prediction-based quality control for approximate accelerators”, *Workshop on Approximate Computing Across the System Stack,* 2015.

[41] Qinfeng Shi, Henry Hoffmann, and Omar Khan, “A HW-SW multicore architecture to tradeoff program accuracy and resilience overheads”, *Computer Architecture Letters,* 2015.

[42] Swagath Venkataramani, Vinay K. Chippa, Srimat T. Chakradhar, Kaushik Roy, and Anand Raghunathan, “Quality programmable vector processors for approximate computing”, In *International Symposium on Microarchitecture,* 1–12, 2013.

[43] Stelios Sidiroglou, Sasa Misailovic, Henry Hoffmann, and Martin Rinard, “Managing performance vs. accuracy trade-offs with loop perforation”, In *ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*. 124–134, 2011.

[44] H. Mahdiani, A. Ahmadi, S. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications”, *IEEE Trans. Circuits Syst. I, Reg. Papers*, Vol. 57, pp. 850-862, 2010.

[45] K. Du , P. Varman and K. Mohanram, "High performance reliable variablelatency carry select addition”, *Proc. Design,Autom., Test Eur. Conf. Exhib. (DATE)*, 2012.

[46] N. Suganthi, “Design and implementation of field programmable gate array based error tolerant adder for image processing application”, *Electronics and Communication Systems(ICECS), 2015 2nd International Conference,* pp.1426-1430, 2015.

[47] N. Zhu, W.L. Goh and K.S. Yeo, &ldquo, “An Enhanced Low-Power High-Speed Adder for Error Tolerant Application”, &rdquo, *Proc. 12th Int',l Symp. Integrated Circuits (ISIC),* pp. 69-72, 2009.

[48] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, “Low-power digital signal processing using approximate adders”, *IEEE Trans. CAD*, 32(1):124–137, 2013.

[49] G. Wallace, “The JPEG still picture compression standard,” *IEEE Trans. Consumer Electron.*, vol. 38, no. 1, pp. xviii–xxxiv, Feb. 1992.

[50] H. Sutter, “The free lunch is over: A fundamental turn toward concurrency in software”, *Dr. Dobb’s journal*, vol. 30.3, pp. 202–210, 2005.

[51] H.A. Mahmoud and M.A. Bayoumi, “A 10-transistor low-power high-speed full adder cell”, *ISCAS'99*, vol. 1, pp. 43-46, 1999.

[52] Honglan Jiang, Jie Han and Fabrizio Lombardi, “A Comparative Review and Evaluation of Approximate Adders”, in *GLSVLSI'15, Proceedings of the 25th IEEE/ACM Great Lakes Symposium on VLSI*, Pittsburgh, PA, USA, 2015

[53] J. Hu and W. Qian. A new approximate adder with low relative error and correct sign calculation. In DATE, in press, 2015.

[54] J. Liang, J. Han, and F. Lombardi, “New metrics for the reliability of approximate and proba- bilistic adders,” *IEEE Transactions on Computers*, 62(9):1760–1771, 2013.

[55] M. Breuer, “Intelligible test techniques to support error-tolerance,” *In Proceedings of the 13th Asian Test Symposium*, pp.386–393, 2004.