Une Famille d'Additionneur Asynchrones CMOS à Temps de Calcul Dépendant de Données

Robin Perrot, Nadine Azemard, Philippe Maurine

Une Famille d'Additionneur Asynchrones CMOS à Temps De Calcul Dépendant Des Données

Robin Perrot, N. Azémard, P. Maurine
LIRMM, UMR CNRS / Université de Montpellier II.
161, rue Ada34392 Montpellier Cedex 5

Email : perrot@lirmm.fr

Résumé

Optimiser un additionneur en conception synchrone repose essentiellement sur une gestion toujours plus fine du calcul des retenues, au prix d'un coût surfacique toujours plus élevé. Ce papier introduit une famille d'additionneurs asynchrones CMOS reposant sur une autre approche : tirer parti de l'influence de la valeur des données sur la durée du calcul. Les architectures proposées ne sont pas fondamentalement nouvelles puisque basées sur les structures d'additions classiques mais elles proposent des performances supérieures à celles proposées par leurs homologues synchrones en s'attachant à la détection des configurations d'opérandes entraînant une simplification des chemins logiques et par conséquent une convergence plus rapide de l'algorithme.

Afin de valider notre approche des opérateurs traitant des opérandes de 16 à 128bits ont été développés sur une technologie CMOS standard 0.35µm et ce, pour différentes contraintes de timing. Les résultats obtenus démontrent que l'exploitation de la dépendance aux données permet d'obtenir des additionneurs nettement plus petits et performants.

1. Introduction

Le caractère 'dépendent aux données' des circuits asynchrones constituant un avantage pour l'obtention de circuits performants, de nombreux travaux ont été dévolus à la conception d'opérateurs arithmétiques asynchrones. Parmi ceux-ci, les additionneurs font l'objet d'un vif intérêt [4-11]. Celui-ci se justifie pleinement puisque l'opération d'addition est mise en jeu par 80% des instructions réalisées par les processeurs [5] et se caractérise par une différence importante entre les temps de calcul moyen et pire cas ($\sim n$ vs $\sim \log_2(n)$ pour un additionneur à propagation de retenue).

Ce temps de calcul variant avec les données impliquées est rendu possible grâce au séquencement du circuit par des signaux de requêtes et d'acquittement (synchronisation dite locale). S'il existe un nombre important de types de circuits asynchrones, il ne subsiste que deux styles différents d'implantation de ces signaux de séquençage. Quelque soit le circuit considéré, celui-ci est nécessairement soit de type micropipelines ou bundled data [1], soit à détection de fin de calcul comme les circuits QDI [2].

Le concepteur d'un circuit à détection de fin de calcul ne fait par définition aucune hypothèse de délai sur les éléments qui le constituent. Il se doit d'utiliser par conséquent des techniques permettant de détecter la fin effective d'un calcul sans équivoque. Parmi celles-ci, on trouve l'utilisation de logique double rail [3], logique DCVSL [4], logique à précharge ou encore des logiques dédiées. Cette approche présente certains avantages. La très faible granularité de la détection de la fin de calcul est l'un d'entre eux. En effet, dans le cas d'une addition à propagation *n* bits, ce type d'approche permet de détecter les *n* valeurs possibles de délai de la structure. Toutefois, le prix d'une telle finesse est élevé. On estime la taille de tels additionneurs de 2 à 6 fois plus élevée que leurs homologues réalisés en logique CMOS simple rail. Ce surdimensionnement apparaît d'autant plus élevé que la complexité du routage associé pénalise fortement la latence directe ou inverse des ces structures. Malgré ces désavantages importants, de nombreuses structures à détection de fin de calcul ont été proposées dans la littérature [4-9].

A l'instar des circuits à détection de fin de calcul, les circuits micropipelines assurent le cadencement des opérations entre les modules les constituant à l'aide de signaux d'acquittement et de requête.

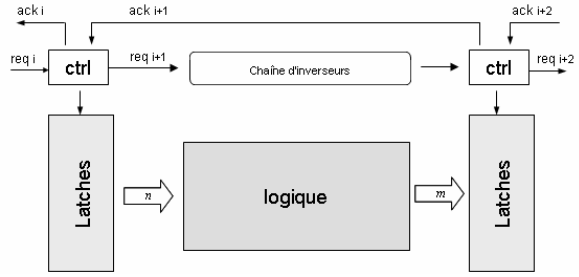

Figure 1 : Un module micropipeline

Ceux-ci s'appuient sur les mêmes hypothèses temporelles que les circuits synchrones : les délais de propagation des modules ont des valeurs bornées et connues. Dès lors, il est possible d'assurer le bon fonctionnement des circuits en adoptant un modèle de délai pire cas pour chacun des modules du pipeline. C'est le rôle d'un second bloc (que nous appellerons chemin de requête) placé parallèlement au traitement des données. Celui-ci ne transporte aucune donnée; il ne fait qu'implémenter -à l'aide d'une chaîne d'inverseurs par exemple- le délai pire cas de la logique combinatoire. Parcouru par les signaux de requête, il les retarde ainsi d'un délai supérieur ou égal au temps nécessaire à la stabilisation des données en sortie, pour n'importe qu'elle condition PVT.

Si cette solution permet d'obtenir des circuits asynchrones de taille équivalente aux circuits synchrones, elle ne permet pas en l'état d'effectuer des calculs en temps moyen. La

chaîne d'inverseurs, et donc le temps mis par une requête pour la parcourir, étant physiquement indépendante des données manipulées. La littérature propose néanmoins une évolution du chemin de requête bundled data traditionnel [10-11] permettant l'obtention de structures d'addition possédant cette propriété. L'évolution utilisée ici est celle que nous appelons implantation à requête bondissante. Elle présente un double intérêt. La structure permettant d'infliger à la requête plusieurs valeurs discrètes de délai est à peine plus complexe qu'un chemin de requête traditionnel ; elle est de plus compatible avec un protocole handshake deux phases

Dans ce contexte, l'objectif de ce papier est de proposer des modifications simples à apporter aux structures d'addition classiques permettant d'obtenir des additionneurs à retenue bondissante performants, c'est à dire proposant une latence réduite pour un faible coût en surface.

2. La requête bondissante : principe

Par définition, les additionneurs à requête bondissante sont des additionneurs asynchrones bundled data (i.e composés d'un chemin de données simple rail et d'un chemin de requête). L'idée sur laquelle ces derniers reposent est de rendre le délai de propagation de la requête à travers ce chemin le plus fortement corrélé possible au temps d'établissement des données sur les sorties du chemin de données.

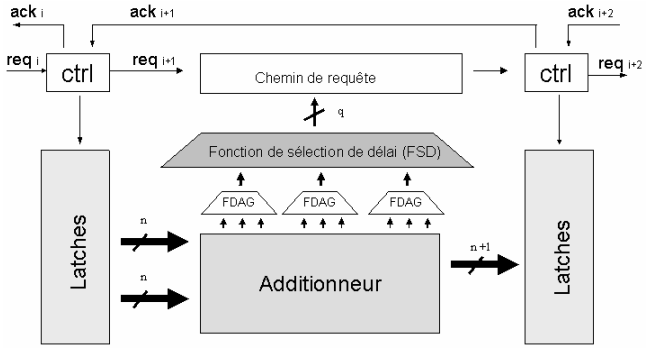

Figure 2 : Module micropipeline à requête bondissante

Leur architecture générique, proposée figure 2, ne diffère de la structure 'bundled data' classique que par la présence de blocs supplémentaires : les FDAG et le FSD. Le rôle des foyers FDAG est de détecter les cas d'absorption et/ou de génération sur une fraction plus ou moins importante de la chaîne de retenue. Leur réponse est interprétée par la FSD qui contrôle le chemin de requête, ou plus précisément le retard à infliger à cette dernière.

Une telle structure permet théoriquement l'obtention d'un chemin de requête pouvant prendre autant de valeurs de délai que le chemin de donnée peut en compter (*n* pour un RCA). Ceci nécessite toutefois une finesse dans la gestion de ces délais difficilement atteignable, ne serait-ce qu'à cause des dispersions du procédé de fabrication. Il est donc plus raisonnable de limiter le nombre de valeurs possibles du délai de propagation du chemin entre 2 et 5 selon la taille des opérandes et l'architecture d'additionneur utilisée. Ceci peut-être réalisé en adoptant comme chemin de requête la structure représentée figure 3. Celle-ci est constituée de chaînes d'inverseurs ou d'éléments de délai (Ti) de multiplexeurs et d'éléments de contrôle (portes de muller par exemple) gérant le protocole handshake.

Une telle implantation n'est pas sans rappeler celle proposée par Nowick et al dans [10] où est introduit le séquençage par "speculative completion". Contrairement à cette dernière, celle-ci ne proscrit toutefois pas l'utilisation d'un protocole 2 phases.

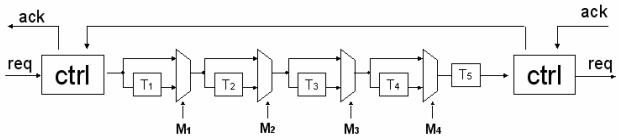

Figure 3 : Implantation possible du chemin de requête

Le fonctionnement de la structure présentée figure 3 s'apparente à celui d'une jauge de la durée de calcul. Lorsqu'une requête est acceptée par l'étage, celle-ci va se propager à travers des éléments de délais et les multiplexeurs. La durée de la propagation de la requête dépend alors du mot binaire M = (M4, M3, M2, M1).

Les enjeux de conception d'un additionneur à requête bondissante sont donc : le choix du nombre de FDAG, leur taille ainsi que leur positionnement le long de la chaîne de retenue.

3. Quelques additionneurs à requête bondissante

Nous introduisons dans cette section trois types d'additionneurs à requête bondissante, basés sur des modifications directes d'architectures d'addition classiques : propagation de retenue (RCA), sélection de retenue (CSA), retenue bondissante (CSK). S'il est possible d'imaginer de nombreuses modifications possibles, celles que nous avons choisies l'ont été en favorisant en considérant deux critères principaux : la simplicité et un impact le plus important possible sur la latence moyenne.

3.1 Additionneur RC à requête bondissante

La figure 4 illustre l'architecture générique d'un additionneur RippleCarry dont on souhaite faire évoluer le temps de calcul selon *u* valeurs différentes. La génération des signaux pilotant les *(u-1)* multiplexeurs du chemin de requêtes s'effectue à partir des signaux délivrés par *(u-1)* foyers de détection répartis le long du RCA de sorte à le fractionner en parties égales (de délai T).

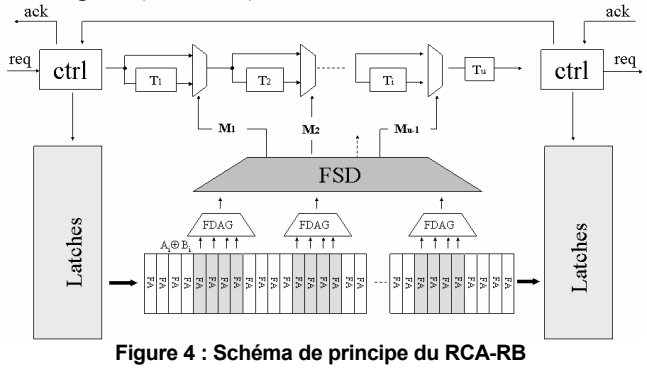

Figure 4 : Schéma de principe du RCA-RB

Les probabilités d'occurrence des *u* valeurs distinctes de délai dépendent de la largeur *l* des foyers de détection. Les valeurs de la latence moyenne *Lm* ont été calculées pour différentes grandeurs de *n*, *u* et *l* en considérant des opérandes aléatoires. Ces calculs montrent que pour *n* et *u* donnés, il existe une valeur *l* minimisant *Lm*. Elles sont répertoriées dans le tableau 1 pour *u* =3.

n	*l*	*Lm* (τ_{FA})
16	*1*	*7.22*
32	*2*	*12.62*
64	*2*	*21.36*
128	*3*	*38.70*

Tableau 1. Valeurs de *l* minimisant la latence moyenne *Lm*

3.2 Additionneur CS à requête bondissante

L'architecture d'un additionneur CarrySelect-RB *n* bits *(n=p.q + r, r<q)* est représentée figure 5. Contrairement à la topologie 'en escalier' de l'additionneur CS classique, celui-ci est divisé en 1 chaîne de *r* FA et *p* chaînes de taille identique, composées de *q* cellules CS. *(p-1)* centres de détection CDAG sont ici disposés sur les *l (q>l>0)* dernières cellules CS de ces chaînes. Le signal qu'ils délivrent révèle que l'attente du signal de sélection du multiplexeur placé au bout de la chaîne considérée est peut-être superflue. En effet, si une absorption ou une génération de retenue se produit sur les *l* dernières cellules, les deux entrées de ce multiplexeur posséderont la même valeur (dans le pire des cas après le délai de *l* cellules CS.)

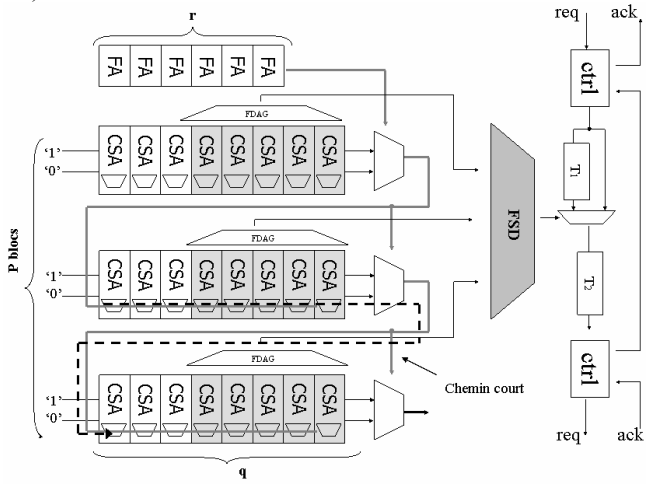

Figure 5 : Schéma de principe du CSA-RB

Le bloc FSD ne réalise ici que le 'et' logique des signaux de sortie des centres de test. La latence d'une telle structure ne peut donc prendre que 2 valeurs distinctes. La première correspond au temps de propagation des *q* cellules d'1 chaîne et d'un multiplexeur (en pointillé), la seconde au temps de propagation de *q* cellules et des *p-1* multiplexeurs.

n	*p*	*q*	*l*	*Lm* (τ_{CSA})
16	*4*	*4*	*4*	*5.46*
32	*8*	*4*	*4*	*7.69*
64	*8*	*8*	*8*	*9.18*
128	*16*	*8*	*8*	*9.97*

Tableau 2. Valeurs de *l* minimisant la latence moyenne *Lm*

Une analyse premier ordre de la latence moyenne nous amène à l'expression :

$$L_m = \left\{1-\left(1-\frac{1}{2^l}\right)^{p-1}\right\}\cdot[q\cdot\tau_{CS}+(p-1)\cdot\tau_{MUX}]+\left(1-\frac{1}{2^l}\right)^{p-1}\cdot[q\cdot\tau_{CS}+\tau_{MUX}] \quad (1)$$

Où τ_{CS} et τ_{MUX} représentent les valeurs respectives des délais d'une cellule CS et d'un multiplexeur. Les valeurs de la latence moyenne *Lm* ont été calculées pour différentes grandeurs de *n*, et *l* en considérant des opérandes aléatoires. Le tableau 2 répertorie les valeurs de *l* minimisant *Lm* pour différentes grandeurs de *n*. Il est à noter que les valeurs optimales sont atteintes pour *l=q*.

3.3 Additionneur CSk à requête bondissante

L'architecture de l'additionneur CarrySkip-RB est présentée figure 6. Celle-ci s'articule autour d'un CSk classique constitué de *p*>2 chaînes 'ripple carry' de dimension *l* et permet à l'obtention de 2 valeurs distinctes de latence. La latence la plus rapide ($3/2.l.\tau_{FA} + \tau_{Mux}$) s'obtient lorsque tous les FDAG détectent au moins un cas d'absorption ou de génération sur 2. *l*/2 successives, la plus lente ($2.l.\tau_{FA} + (p-1).\tau_{Mux}$) s'obtenant dans le cas contraire.

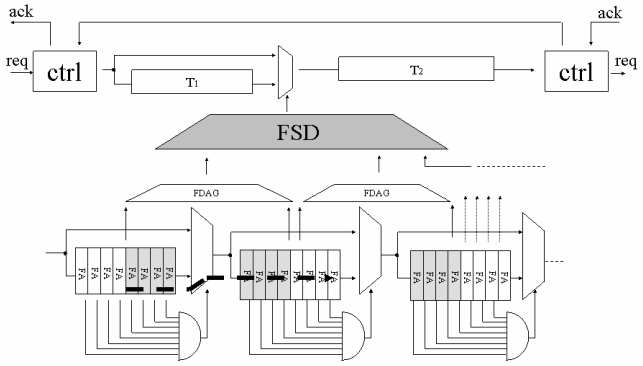

Figure 6 : Schéma de principe du CSkA–RB

Les probabilités d'occurrence de ces deux latences varient fortement avec les valeurs de *l* et *p*. Une analyse premier ordre de la latence moyenne nous amène à l'expression :

$$L_m = \left\{1-\left(1-\frac{1}{2^l}\right)^{p-1}\right\}\cdot[2\cdot1\cdot\tau_{FA}+(p-1)\cdot\tau_{MUX}]+\left(1-\frac{1}{2^l}\right)^{p-1}\cdot\left[\frac{3}{2}\cdot1\cdot\tau_{FA}+\tau_{MUX}\right] \quad (2)$$

Celle-ci indique que cette latence moyenne devient proportionnelle à *n/p* dès que *l=n/p* >4. Le tableau 3 répertorie les valeurs de *l* minimisant *Lm* pour différentes grandeurs de *n*.

n	*p*	*l*	*Lm* (τ_{FA})
16	*4*	*4*	*7.70*
32	*8*	*4*	*9.91*
64	*16*	*8*	*13.27*
128	*16*	*8*	*14.03*

Tableau 3. Valeurs de *l* minimisant la latence moyenne *Lm*

4. Performances

À des fins de validation, ces additionneurs ont été implantés sur une technologie CMOS standard 0.35µm et ce, pour différentes contraintes de timing.

Afin de limiter au maximum le surcoût en surface du à la présence du chemin de requête et des FDGA, on aura ici soin de n'optimiser que les chemins de propagations de données dits 'courts' et de laisser intacts les chemins générant une latence plus importante. Si la latence pire cas n'est par conséquent plus optimale, sa très faible probabilité d'occurrence garantit une utilisation optimisée du silicium au prix d'une dégradation minimale de la latence moyenne.

Les performances proposées par ces structures (additionneur + FDAG + DSF + chemin de requête) obtenues après synthèse et placement routage pour 10 000 vecteurs aléatoires, sont reportées en même temps que leurs homologues synchrones sur les figures 7, 8 et 9. Ces graphiques reportent en abscisse leur surface tandis que l'axe des ordonnées présente leur latence moyenne *Lm,* et ce pour des opérandes

variant de 32 à 128bits. Les aires ainsi décrites représentent par conséquent l'espace de conception disponible pour le designer, pour un nombre de foyer de détection fixé (3 pour le RB-RCA, et définit par *n* pour les autres structures). (Notons que cet espace n'est pas en réalité trapézoïdal, car seul l'additionneur RCA présente un délai directement proportionnel à la taille de ces données, il n'en demeure pas moins que cette première approximation reste représentative.)

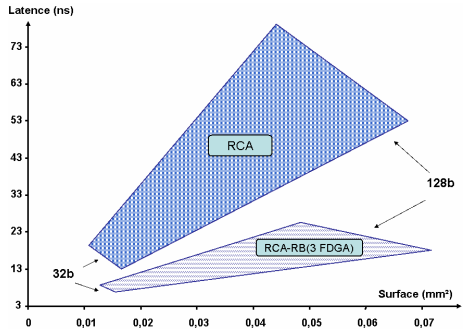

Figure 7 : Comparaison des performance RCA vs RCA-RB

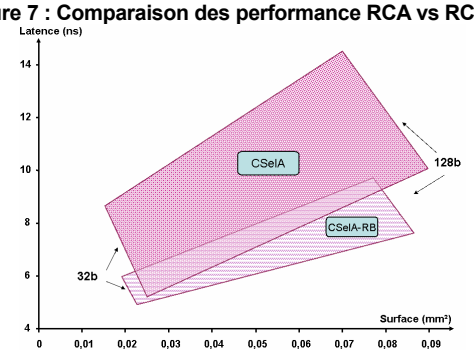

Figure 8 : Comparaison des performance CSA vs CSA-RB

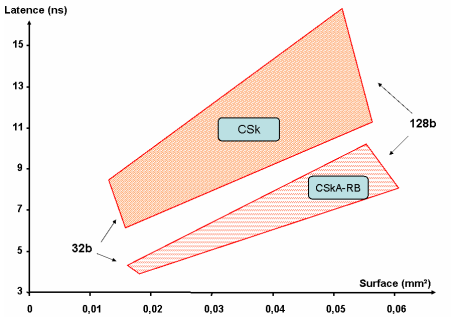

Figure 9 : Comparaison des performance CSK vs CSkA-RB

Ces représentations montrent que l'approche requête bondissante présente un espace de conception permettant d'atteindre des latences plus faibles pour un même coût en surface. (~70% de réduction pour un RCA 128bits).

Les performances de ces additionneurs RB ainsi que la structure présentée dans [10] sont résumées figure 10. Comme cela est illustré, la structure promettant le meilleur compromis latence – surface est celle du CSkA-RB. Il est aussi démontré que l'approche 'requête bondissante' s'avère plus scalable que celle décrite dans [10], qui voit son coup en surface devenir fortement pénalisant à mesure que la taille des opérandes augmente.

Une estimation complète de la consommation de ces structures reste à ce jour à réaliser. Les premiers résultats tendent tout de même à montrer que l'éventuel surcoût dû à la logique supplémentaire est compensé par l'absence d'optimisation des chemins 'longs', qui ne sont ni dimensionnés, ni bufferisés. Le bilan énergétique pour effectuer un nombre d'addition donné demeurant nettement en faveur des additionneurs asynchrones (~20% pour 100 additions aléatoires par RCA-RB).

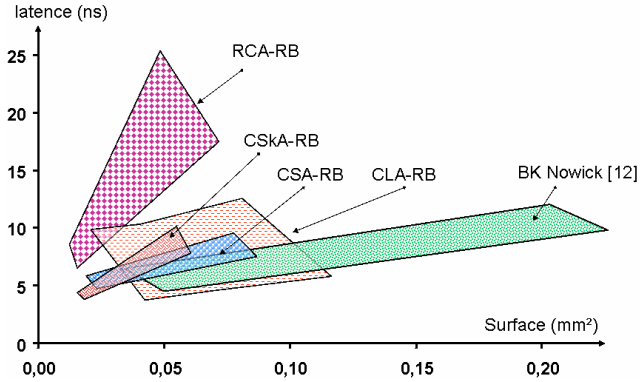

Figure 10 : Comparaison des performance d'additionneurs RB

5. Conclusion

Nous avons introduit des architectures d'addition asynchrones à temps de calcul fortement dépendant des opérandes traitées. Obtenus à moindre coût surfacique à partir des structures d'addition classiques, elles se distinguent des structures d'additionneurs data dependent déjà existantes par l'utilisation de logique CMOS standard. Leur implantation est par conséquent très simplifiée par rapport à l'utilisation de logique cascode ou double rail qui pénalise généralement les performances en vitesse et consommation. Les résultats obtenus démontrent que l'exploitation de la dépendance aux données permet d'obtenir des additionneurs nettement plus petits et performants.

Références

[1] I.E Sutherland, "Micropelines" Commun. ACM, vol 32, 1989

[2] A.J. Martin "Asynchronous datapaths & the design of asynchronous adders" Formal Methods in systems design, vol1:1 1992.

[3] A.W. Burks, H.H. Goldstine, J. Von Neumann, "Preliminary discussion of the logical design of an electronic computing instrument", in Bell & Newell, Computer structures; readings & examples, computer sciences series, Mc Graw-Hill, 1971.

[4] H.Dhanesha, A. Albicki, "Self-timed adder with pipelined output". Proc.of the 36th Midwest Symposium on Advanced Research in Asynchronous Circuits & Systems, 1993.

[5] J.Garside, "A CMOS VLSI Implementation of an Asynchronous ALU", IFIP Working Conference on Asynchronous Design Methodologies, 1993.

[6] M. Renaudin, B. El-Hassan, "The design of fast asynchronous adders and their implementation using DCVSL logic" IEEE International Symposium on Circuits and Systems, 1994.

[7] A.D. Gloria, M.Olivieri "Statistival Carry lookahead Adders", IEEE trans. on computers, vol.45, 1996.

[8] J.M Muller, A.Tisserand, J.M Vincent "Asynchronous sub-logarithmic adders", IEEE Pacific Rim Conference, 1997.

[9] G.A. Ruiz, "Evaluation of three 32-bit CMOS adders in DCVS logic for self-timed circuits". IEEE Journal of Solid-Sate Circuits, Vol.33, Issue 4, April 1998.

[10] S.M Nowick, K.Y. Yun, P.A Beerel "Speculative completion for the design of high performance asynchronous adders", 3rd ASYNC symposium, 1997.

[11] R. B Reese, M. A Thornton, C. Traver, "Arithmetic logic circuits using self-timed bit level dataflow & early evalutation" Proceedings of ICCD: VLSI in Computers & Processors 2001.