RC-005

デジタル補聴器用 DSP を対象とした非同期式直列加算器
An asynchronous serial adder for DSP hearing aid

近藤 真史[†] **岡本 大地**[‡] **横川 智教**[‡] **有本 和民**[‡] **佐藤 洋一郎**[‡]
Masafumi Kondo Daichi Okamoto Tomoyuki Yokogawa Kazutami Arimoto Yoichiro Sato

1. まえがき

近年，高齢化社会の進展に伴って，日本の推定難聴者は約 2000 万人に達しており，デジタル信号処理回路 (Digital Signal Processor：DSP) を内蔵した，いわゆるデジタル補聴器が広く普及している [1]．最近のデジタル補聴器は様々な聴覚支援機能を備えているが，相対的にそれを処理する DSP への演算負荷が増大し，デジタル補聴器の電池寿命は数日程度に留まっているのが現状である [2]．特にデジタル補聴器はその装着形態から搭載可能な電池に物理的な限界があるため，電池寿命の改善を図るには，デジタル補聴器用 DSP(以下，単に補聴器用 DSP という) におけるアーキテクチャの観点から低消費電力化を図る必要がある．

一般的な DSP は，様々な信号処理を実現するために高速な積和演算器を内蔵しているが，補聴器用 DSP が処理する音声信号は高々20kHz 程度であるため，高速な演算器は冗長となる．そこで，現在常識的に用いられている並列乗算器ではなく，n ビット加算器とシフトレジスタを用いて逐次的に乗算処理を行う，いわゆる直列乗算器を補聴器用 DSP に応用する試みが検討されつつある [3]．特に文献 [4][5] のそれはクロックを用いない非同期式回路として設計されており，補聴器用 DSP を指向した低消費電力な乗算器を実現している．一方，これらの非同期式直列乗算器や積和演算器を構成する加算器には，依然として n 個の全加算器 (FA) を用いた並列加算器が用いられている．したがって加算器についても同様に，一対の FA とフリップフロップ (D-FF) のみを用いて逐次的に加算処理を行う直列加算器を採用し，その非同期化を図ることで，さらに低消費電力な演算器の実現を期待できる．

そこで本論文では，補聴器用 DSP に内蔵される演算器への応用を前提とした非同期式直列加算器 (以下，単に非同期加算器という) の構成法を提案する．この種の非同期式回路では，タイミング制御に係る休止相の挿入に伴って信号の遷移回数が増加し，演算性能の劣化や動的な消費電力の増大を招く可能性がある．これについて本論文では，さらに一対の FA と D-FF を並列に設け，それらを相補的に制御することにより，休止相に起因する無効な信号遷移を隠蔽する手法を採る．そして，非同期加算器の設計結果を示すとともに，既存の並列加算器との性能比較を通じてその有効性を示す．

2. 設計方針

直列加算器の非同期化を図るには，クロックに代わって動作タイミングを制御する手法が必要となる．以下，対象となる動作タイミングとその制御手法を述べる．

1 ビット加算の完了　直列加算器では，FA により 1 ビットの加算が完了したタイミングで，加数，被加数および和が格納されているシフトレジスタのデータをシフトする必要がある．非同期式回路においてこの種のタイミング制御手法は，束データ方式と 2 線 4 相式に大別されるが，本論文では，補聴器用 DSP への適用を前提としていることから，環境耐性に優れ，回路遅延に応じた最適な動作を実現可能な 2 線 4 相式を採用する [6]．2 線 4 相式では，1 ビットの信号 (d) を表すため 2 ビットの信号 (d_1, d_0) を用いる．(d_1, d_0) が (0, 1) の状態は論理 ‘0’，(1, 0) は論理 ‘1’ を表し，(0, 0) は無効な信号を表す．特に，信号線上に有効なデータが存在する状態を稼動相，存在しない状態を休止相という．そして，稼動相と休止相を交互に繰り返すことにより，各相の境界を以って処理の完了を検出できる．

n ビット演算の完了　上述の制御に基づいて n ビットの加算処理を実現するためには，n ビット目の加算が完了したタイミングを検出し，それに応じて以降の動作を停止する必要がある．一つには加算回数を数えるためのカウンタを別途設ける手法も考えられるが，一般的に非同期式カウンタの正確な制御は困難である上に，回路面積が著しく増大することとなる．そこで本論文では，演算終了までに演算結果を格納するレジスタ (後述の SRS) が n 回シフトされる点に着目し，SRS を最上位ビットのみが ‘1’ である信号列 “10...0” で予め初期化しておくことにより，最下位ビットに ‘1’ が現れたタイミングで演算の終了を検出する方針を採る．

3. 非同期式直列加算器

3.1. 構成

提案する非同期加算器の構成を図 1 に示し，各構成要素の機能と意味を以下に記す．また，ラベル C が付記された論理ゲートはマラーの C 素子であり，入力が共に ‘1’ ならば ‘1’，‘0’ ならば ‘0’，それ以外の場合は直前の出力を保持する回路である．

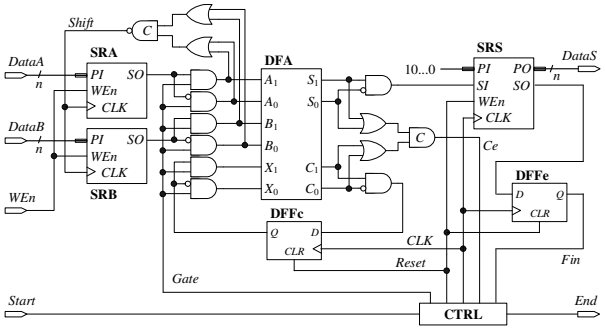

図 1: 非同期加算器の構成

[†]川崎医療福祉大学 医療技術学部
[‡]岡山県立大学大学院 情報系工学研究科

SRA，SRB および SRS：加数 ($DataA$)，被加数 ($DataB$) および和 ($DataS$) を格納するシフトレジスタ．なお，SI および PI は直列入力および並列入力，SO および PO は直列出力および並列出力であり，WEn のアサートにより入力されているデータ (PI) を保持し，CLK の立上りエッジでデータをシフトする．
DFA：文献 [7] に基づく 2 線式全加算器．なお，X および C はそれぞれ加算前後における桁上げであり，A, B および S と同様に 2 線符号化が施されている．
DFFc：演算過程で生じた桁上げを保持する非同期クリア (CLR) 付の D-FF．
DFFe：SRS の最下位ビットから出力される演算の完了を表す信号 End を保持するための D-FF．
CTRL：各シフトレジスタのシフトタイミングを制御するコントローラ．

3.2. 動作手順

SRA および SRB に対するデータ格納から演算の完了に至るまでの動作手順は以下の通りである．
手順 1．演算データの格納　WEn がアサートされると，SRA および SRB に $DataA$ および $DataB$ がそれぞれ格納される．
手順 2．制御信号の生成　WEn のネゲート後，演算の開始を指示する信号 $Start$ がアサートされると，CTRL は $Gate$ をアサートする．
手順 3．1 ビット加算の開始　$Gate$ がアサートされることで，SRA および SRB の最下位ビットのデータが 2 線符号化されて DFA に入力され，稼働相が動作する．
手順 4．和と桁上げの格納　加算が終了し，桁上げ S と和 C が共に安定したことを C 素子により検出すると，Ce がアサートされる．これを受けた CTRL は CLK をアサートし，再び 1 線式のデータに変換された S および C をそれぞれ SRS および DFFc に格納する．
手順 5．入力レジスタの制御　CLK のアサートに続けて CTRL は $Gate$ をネゲートする．これに伴って DFA の入力全てが ‘0’ となるため，DFA に休止相が動作するとともに，図 1 上部の C 素子により $Shift$ が アサートされ，SRA と SRB のデータがシフトされる．
手順 6．1 ビット加算の終了　DFA において休止相が終了すると Ce がネゲートされる．これを受けた CTRL は，CLK をネゲートするとともに，再び稼動相を動作させるために $Gate$ をアサートし，手順 3 に戻る．
手順 7．演算処理の完了　手順 3〜手順 6 の動作を n ビット分繰り返すことにより，演算結果が順に SRS に格納される．それに伴って，SRS の初期値として最上位ビットに格納していた信号 ‘1’ が DFFe に格納され，演算の完了を表す信号 Fin としてアサートされる．そして，Fin を受けた CTRL は End をアサートして演算の完了を通知し，非同期加算器はその動作を停止する．

3.3.CTRL の設計と構成

各レジスタを制御する CTRL は非同期式順序回路として設計する必要がある．本論文では，CTRL の動作が信号遷移のみで規定できる点に着目し，マラー回路に基づいて設計を行う．マラー回路の設計には，回路の動作を記述した信号遷移グラフ (STG) が用いられ，これを入力として論理合成を行うことができる [8]．図 1

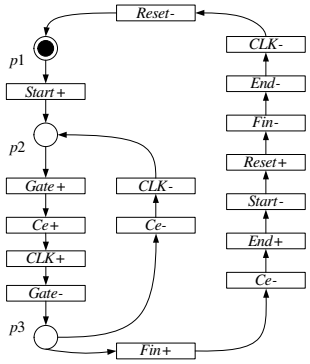

図 2: CTRL の STG

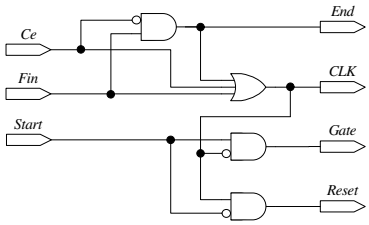

図 3: CTRL の回路構成

の回路構成および 3.2 の動作手順に基づいて CTRL の動作を表現した STG を図 2 に示す．図中の○は状態，□は信号遷移 (+は信号の立上り，−は立下り)，矢印は遷移関係を表し，現存の状態は●を付して表される．

まず，初期状態 p_1 に対して $Start$ がアサートされると p_2 に遷移し，さらに続けて稼働相に係る信号 $Gate$ をアサートする．次に，DFA による加算を経て Ce がアサートされると，CLK をアサートするとともに $Gate$ をネゲートし，状態 p_3 に遷移する．これ以後の遷移は，演算が終了するか否か，すなわち Fin がアサートされるかに依存する．これがアサートされることなく Ce がネゲートされた場合は，CLK をネゲートし，再び状態 p_2 に戻って演算を継続する．一方，Fin がアサートされた場合は，Ce がネゲートされるまで待機した後に End をアサートする．そして，演算結果の退避後に $Start$ がネゲートされると，SRS および各 D-FF の初期化に係る信号 $Reset$ をアサートする．これにより DFFe が初期化，すなわち Fin がネゲートされるため，End および CLK をネゲートして状態 p_1 に戻る．

非同期式回路合成ツール Petrufy4.2 [8] に上述の STG を入力して得られた CTRL の回路構成を図 3 に示す．

4. 休止相の隠蔽による高速化

上述の非同期加算器は，単一の DFA を用いて稼動相と休止相を交互に繰り返すことで演算を行うが，休止相において加算処理自体は行われていない．この休止相に起因する無効な信号遷移は，非同期加算器を乗算器へ応用した場合における演算時間へのオーバーヘッドや動的な消費電力の増加を招くことが容易に推測される．これがこれまでに直列加算器の非同期化が検討されていない要因の一つと考えられる．

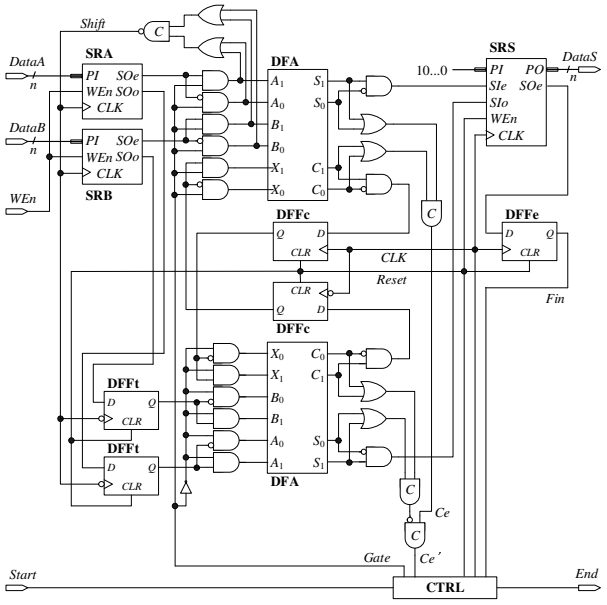

図 4: 隠蔽型加算器の構成

そこで本論文では，稼動相と休止相が交互に処理される点に着目し，二対の非同期加算器を用いて相補的かつ交互に演算を行うことにより，休止相に起因する無効な信号遷移を隠蔽する手法を提案する．以下，これに基づいた非同期加算器を隠蔽型加算器といい，その構成を図 4 に示す．基本的な回路構成は図 1 の非同期加算器を上下対称に配置した形態を採るが，下段のそれに対しては，CTRL からの制御信号が反転入力されている点，データの入力タイミングの整合性を図るための DFFt が挿入されている点に注意する．また，上下段で排他的に処理される各相のタイミングを合わせるため，各非同期加算器から出力される Ce 信号を C 素子により同期化し，改めて Ce' 信号として生成している．この構成によれば，上下段の非同期加算器でそれぞれ稼動相と休止相が交互に動作し，常にいずれかの非同期加算器において稼動相が処理される．

ここで，上述の処理を実現するためには，上段の非同期加算器に i $(0 \leq i < n)$ ビット目のデータ，下段に $i+1$ ビット目のデータをそれぞれ異なるタイミングで DFA へ入力する必要がある．より具体的には，上段の非同期加算器には偶数ビット目 $(i = 0, 2, 4, ..., n-2)$ のデータ，下段には奇数ビット目 $(i = 1, 3, 5, ..., n-1)$ のデータをそれぞれ相補的にアサートすることとなる．この制御について本論文では，CTRL の構成を変更することなく，入出力シフトレジスタの配線のみを変更することで簡便に実現する方針を採る．隠蔽型加算器に用いるシフトレジスタの構成を図 5 に示す．図中の直列入出力信号 SI/SO の末尾に付記された e および o は，それぞれ偶数ビット目および奇数ビット目のデータを表す．図 5 より，隠蔽型加算器に用いるシフトレジスタは，i と $i+1$ 番目の D-FF をカスケード接続するのではなく，i と $i+2$，$i+1$ と $i+3$ 番目の D-FF を接続し，特に後者の D-FF に対しては CLK を反転して入力する．これにより，CLK の立上りで偶数ビット目，立下りで奇数ビット目のデータがそれぞれ整合性を失うことなくシフトすることができる．

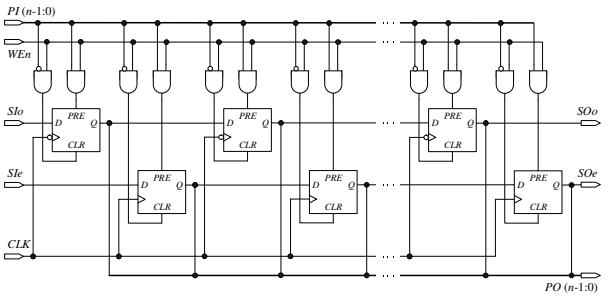

図 5: 隠蔽型加算器に対するシフトレジスタの構成

5. 設計と評価

5.1. 動作確認と演算性能

提案した非同期加算器と隠蔽型加算器を Xilinx ISE14.6 により設計し，同社製 FPGA Virex5-LX30 を対象としたタイミングシミュレーションを通してその動作確認を行った．その一例として，$DataA$ = “00001111”，$DataB$ = “01010101” とした場合におけるシミュレーション結果を図 6 に示す．各信号の意味は図 1 および図 4 と同様であるが，非同期加算器と隠蔽型加算器のそれを区別する必要がある場合には，信号名の末尾にそれぞれ $\_a$ および $\_c$ を付している．図 6 より，$Start$ のアサート (200ns) に伴って演算に係る信号群が遷移し，演算が完了すると 290ns および 250ns 付近でそれぞれの End がアサートされている．このとき，いずれも演算結果として $DataS$ = “01100100” が出力され，適切に演算が行われていることを確認できる．

次に，n=8, 16, 32, 64 における非同期加算器と隠蔽型加算器の設計を行った．また，比較対象として 4 ビットを単位とした桁上げ先見加算器も同様に設計を行った．これらの加算器についてその演算時間を解析した結果を図 7 に示す．ここで，非同期加算器と隠蔽型加算器の演算時間は $Start$ のアサートから End がアサートされるまで，桁上げ先見加算器はデータの入力から演算結果が確定するまでとした．図 7 より，上下段の非同期加算器の同期化に係るオーバーヘッドの影響があるものの，隠蔽型加算器の演算時間は非同期加算器のそれに比して約 40%短縮されており，休止相の隠蔽効果を確認できる．ここで，n 回の加算を繰り返す直列乗算器に対して，最も演算時間を要する 64 ビット非同期加算器の応用を想定した場合，その演算時間は $730.753[ns] \times 64 = 46.768[\mu s]$ となる．これは，市販のデジタル補聴器のサンプリング周波数が 20 kHz 程度である点 [9] を勘案すると，そのサンプリング周期 $50[\mu s]$ を満たしている．さらに一般的な音声信号の量子化ビット数が 16 ビットである点を考慮すると，提案の加算器はいずれもデジタル補聴器用 DSP の演算器へ応用する上で十分な演算性能を備えている．特に，原理的に優れた演算性能を実現できる並列加算器に比して，隠蔽型加算器の演算時間が 20%程度の劣化に留められているのは非同期化の特筆すべき点である．

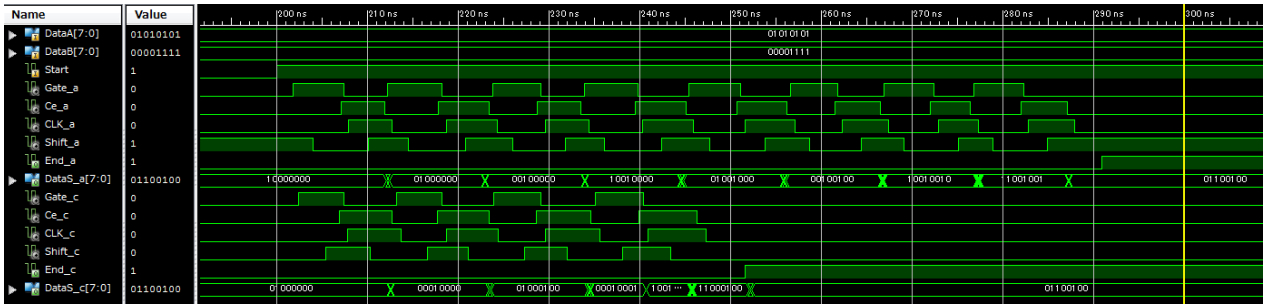

図 6: シミュレーション結果

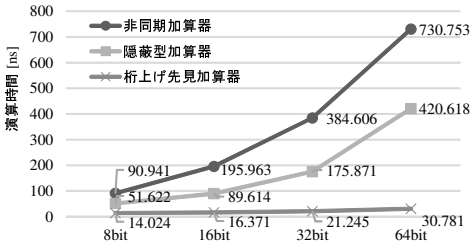

図 7: 各加算器の演算時間

5.2. 消費電力

Xilinx XPower Analyzer を用いて FPGA 実装時における消費電力と回路面積の解析を行った結果を表 1 に示す．なお，各加算器の条件を統一するため，桁上げ先見加算器の評価結果には入出力レジスタを含んでいる．まず，回路面積 (LUT および FF) に着目すると，提案の加算器の面積は n に依存しないにも関わらず，いずれの加算器も同様の資源量と傾向を示している．これは，本論文の加算器では並列入出力機能付のシフトレジスタを利用しており，これを制御するマルチプレクサのオーバーヘッドによるものと考えられる．一方，消費電力については，提案の加算器はいずれも桁上げ先見加算器に比して低消費電力に動作し，特に隠蔽型加算器のそれは約半分となっている．これは，隠蔽型加算器のシフトレジスタが奇数ビット目と偶数ビット目を独立に管理しているために，各ビットのデータが経由する D-FF の数が半減し，その信号遷移に係る動的な消費電力が低減された結果と考えられる．特にこれら提案加算器の低消費電力性の効果は，直列乗算器に応用した際により顕著に現れるものと考えられる．

6. あとがき

本論文では，補聴器用 DSP に対する非同期式直列加算器の構成法を提案した．特にこの加算器では，二対の全加算器を相補的に制御することにより，休止相に起因する演算時間のオーバーヘッドを隠蔽し，補聴器用 DSP へ応用する上で十分な演算性能を実現している．そして，シミュレーションを通じて所望の動作を確認するとともに，既存の並列加算器に比して消費電力を約 50%削減できることを確認している．今後の課題としては，提案の加算器を乗算器や積和演算器に応用し，その有効性を確認することが挙げられる．

謝辞 本研究の一部はウエスコ学術振興財団「平成 26 年度学術研究費助成」の支援を受けて実施された．

表 1: 各加算器の消費電力と所要面積

	n	消費電力 [mW]	LUT	D-FF
桁上げ先見加算器	8	0.42	25	24
	16	2.04	61	48
	32	4.08	133	96
	64	7.28	277	192
非同期加算器	8	0.18	63	26
	16	0.52	95	50
	32	1.74	159	98
	64	6.65	290	194
隠蔽型加算器	8	0.25	89	29
	16	0.52	122	53
	32	1.20	187	101
	64	3.51	316	197

参考文献

[1] 日本補聴器工業会, “補聴器供給システムの在り方に関する研究,” 2 年次報告書, 2003.
[2] J. DiCristina, “Introduction to Hearing Aids and Important Design Considerations,” Maxim integrated, 2010.
[3] K. C. Killpack, et al, “A standard-cell self-timed multiplier for energy and area critical synchronous systems,” Proc. ARVLSI 2001, pp.188-201, 2001.
[4] D. Kearney, et al, “Bundled data asynchronous multipliers with data dependent computation times,” Proc. ASYNC 1997, pp.186-197, 1997.
[5] G. Bah-Hwee, et al, “A Low-Voltage Micropower Asynchronous Multiplier With Shift-Add Multiplication Approach,” IEEE Trans. Circuits and Systems, Vol.56, No.7, pp.1349-1359, 2009.
[6] 齋藤 寛, “非同期式回路の設計技術,” IEICE Fundamentals Review, Vol.3, No.3, pp.64-70, 2009.
[7] 今井 雅, 他, “非同期式乗算器の設計と試作,” 信学技報 ICD, Vol.96, No.20, pp.33-40, 1996.
[8] J. Cortadella, et al, “Petrify: A Tool for Manipulating Concurrent Specifications and Synthesis of Asynchronous Controllers,” IEICE Trans. Information and Systems, Vol.80, No.3, pp.315-325, 1997.
[9] H. McDermott, “SoundRecover : The importance of wide perceptual bandwidth,” Phonak. Background Story, 2010.