ソフトエラー耐性を有するデータパス合成における最適化について

井上 恵介[1,a)]

概要：本論文ではソフトエラー耐性を有する ASIC の高位合成問題について議論する。特にデータパスの記憶回路であるレジスタに着目し、ECC によるエラー訂正を行う設計問題を考える。データの安全性の尺度となるセーフタイムという概念を導入し、セーフタイムに基づいて ECC モジュールを最小化しつつソフトエラー耐性を有するデータパス設計を提案する。整数計画法による解法を提案し高位合成のベンチマーク回路に適用した結果を示す。

On the Optimization of Soft Error-Tolerant Datapath Synthesis

Inoue Keisuke[1,a)]

Abstract: This paper discusses high-level synthesis of soft error-tolerant ASIC. Specially, we focus on registers (memory part of data-path), and consider a design problem of error correcting by ECC. We introduce a new concept of safe-time, and propose a soft error-tolerant data-path design with minimizing the number of ECC modules based on this concept. We propose an integer linear programming-based approach to this design problem, and show a case study to apply the approach to a benchmark circuit.

1. はじめに

シリコン・プロセス技術の進展に伴い、半導体デバイスのサイズが急速に縮小している。それによって、集積回路は性能向上、低消費電力化、集積度向上、製造コストの削減を同時に達成してきた [1]。その一方で、新たな問題が顕在化している。宇宙からの放射線などの影響により回路中の論理値が反転し結果として回路の出力が誤る問題である。縮耐故障などの永久故障とは異なり、正常な回路にも一時的に出現するためソフトエラーとよばれる。動作周波数の増大、負荷容量の減少、供給電源電圧の低下によって、ソフトエラーの問題は将来に渡って深刻な問題になると考えられる [2]。たとえ、1 つのトランジスタのソフトエラーレートを削減しても、総トランジスタ数が増加しているのでチップ全体のソフトエラー率は増大する。

レジスタのソフトエラー問題の対策手法は大別して二つある。一つ目の手法はエラー検知のために冗長ビットをデータに付加する手法（パリティビット）である。パリティビットは消費電力や面積面で低コストであるが、エラーを訂正する能力はない。二つ目の手法は ECC (エラー訂正符号) に基づく手法である。ECC はエラー検知のみならず、複数ビットの誤り訂正が可能である。近年、複数ビットが反転するタイプのレジスタのソフトエラーの発生も報告されており [3]、ECC は強力な手段である。その代りパリティビットと比較すると消費電力や面積のコストが大きいといった欠点がある [4]。

特定の用途向けに様々な機能を有する回路を 1 つにまとめた LSI を ASIC (Application Specific Integrated Circuit) とよぶ。ASIC はデータパス回路とコントローラ回路からなるが、本稿ではそのうちのデータパス回路の設計問題に着目する。特定用途向けアーキテクチャの性能を向上させるために ASIC の複雑度は日々増大している。データパス回路は実際に計算を行う演算器、データを記憶するレジスタ、複数の入力から一つの入力を選択して出力するマルチプレクサ、データを転送する経路であるワ

[1] 金沢工業高等専門学校
Kanazawa Technical College, Hisayasu 2-270, Kanazawa 921–8164, Japan
[a)] k-inoue@neptune.kanazawa-it.ac.jp

ⓒ 2016 Information Processing Society of Japan

イヤから構成される。ASIC のデータパス回路のソフトエラーに関する先行研究では主に演算器におけるソフトエラー、または回路の一部分 (演算器、レジスタ、マルチプレクサを含む) のソフトエラーの影響に注目しているが、レジスタのみに着目した研究は著者の知る限り存在しない [5], [6], [7], [8], [9], [10], [11], [12], [13]。レジスタが ASIC 全体に占めるコストの割合 (面積や消費電力) はデータパス回路全体の中では小さいが、レジスタは ASIC の動作中に頻繁にアクセスされるため、レジスタに発生するソフトエラーの影響は ASIC 全体に急速に伝搬していき誤った出力結果につながると考えられる。

本稿では、レジスタのソフトエラーに耐性を持つデータパス合成のために ECC に基づく新しい手法を提案する。本稿で報告する結果を以下にまとめる。

- **セーフタイム**という新しい概念を導入する。セーフタイムはデータの安全性 (ソフトエラーに対してデータの正しさが期待されること) をクロックサイクル数で定義する。セーフタイムの間は ECC モジュール (ECC に基づいてエラー検知および修正を行うモジュール) をデータに適用する必要がない。その分、電力の節約が期待できる。
- **ECC スケジューリング**という新しい高位合成タスクを導入する。これは、いつどのデータに ECC を適用するかを決定するタスクである (本稿では簡単化のため、ECC を用いたエラー検知および修正のことを、単に「ECC を適用する」という)。
- 一つのステップで ECC を適用できる最大数はデータパスに実装された ECC モジュール数で決定される。回路コストを削減するためには ECC モジュール数を最小化することは重要である。異なる ECC スケジューリングは異なる最小 ECC モジュール数をもたらすため、ECC スケジューリングを最適化し ECC モジュール数を最小化する問題を定式化する。
- 整数計画法に基づく解法を提案し、ケーススタディとして高位合成のベンチマーク回路に適用した結果を示す。

2. ソフトエラーに耐性を有する設計の問題

2.1 ECC に基づくレジスタシステム

図 2 は本稿で想定する ECC モジュールを付加したレジスタのブロック図である。ECC 適用時、レジスタの出力は ECC モジュールの入力に印加され、エラー修正されたデータが再びレジスタに書き込まれる。ECC の適用の有無はコントローラの制御信号によって制御する。

本稿では以下の仮定を採用する。(1) 生成直後のデータはソフトエラーフリーである (レジスタ以外で発生したソフトエラーは他の手法でカバーされる)。(2) レジスタにソフトエラーが発生する確率は時間に比例する。

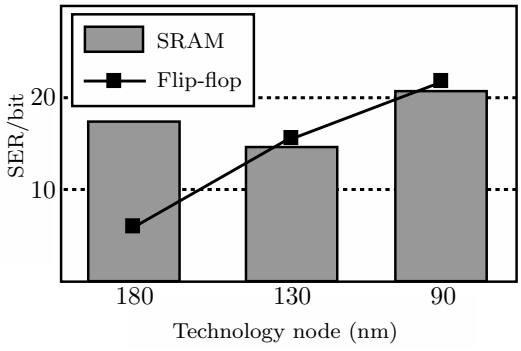

図 1 Scaling trend for alpha-induced soft error rate (SER) of flip-flops.

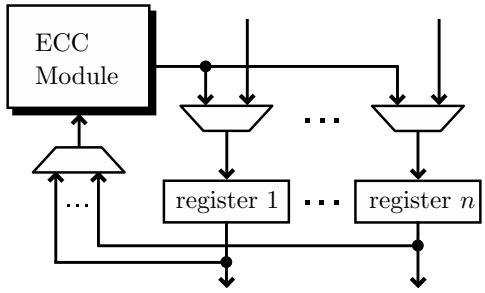

図 2 Block diagram of registers with an ECC module.

2.2 数学モデル

高位合成の入力であるデータフローグラフ (data flow graph: DFG) を無閉路有向グラフ $G(O, A)$ で定義する。ここで、$O = \{o_i \mid i = 1, \ldots, n\}$ は頂点の集合であり、頂点は演算を表す。A は演算間のデータの依存関係を表す。演算 o_i の出力データを d_i. で表す。時間軸はクロック信号の立ち上がりエッジで分割され、各分割時刻をステップとよぶ (j 番目のステップをステップ j とよぶ)。演算スケジューリングは演算をステップに割り当てる高位合成タスクで演算結果をレジスタに記憶させるステップを表している。図 3(左) は例題 DFG とその演算スケジューリング結果を表している (例えば、演算 o_3 はステップ 2 に演算スケジューリングされている)。本稿では問題の簡単化のため演算スケジュールは設計の入力として与えられていると仮定する。データ d_i のライフタイムは o_i が演算スケジューリングされたステップから最後にそのデータが他の演算によって参照されるステップまでの時間的開区間で定義する。例えば図. 3 ではデータ d_1 のライフタイムはステップ 1 から 4 までの開区間で定義される。レジスタの許容ソフトエラー率 (Permissible Soft error Rate: PSR) が設計条件として与えられるものとする。仮定により、生成直後のデータはソフトエラーフリーでありレジスタにソフトエラーが発生する確率は時間に比例する。ソフトエラー発生確率を表す確率変数も設計条件の一部として与えられていると仮定する。この情報を基に、ECC の適用をしなくてもデータの安全性を保持できるステップ数を計算できる。そのようなステップ数を**セーフタイム**と呼び X で表す。

ⓒ 2016 Information Processing Society of Japan

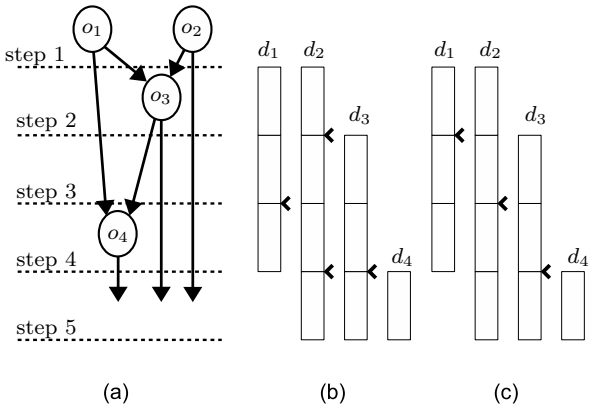

図 3 (a) Example scheduled DFG. (b) An ECC scheduling result requiring two ECC modules. (c) An ECC scheduling result requiring one ECC module.

2.3 最適化問題の動機となる例題

本校では新しい高位合成タスクである ECC スケジューリングを導入する。ECC スケジューリングはどのデータにどのステップで ECC を適用するかを決定するタスクである。本章では、異なる ECC スケジューリングの結果が異なる最小 ECC モジュール数を必要とすることを図 3(a) に示す例題で示す。セーフタイムとして $X = 2$ と仮定する (データが生成されてから、もしくは最後に ECC を適用してから 2 ステップ以内にデータに ECC を適用する必要があるという意味)。利用可能な ECC モジュールは 1 個とする。図 3(b) では、d_1 のライフタイム長がセーフタイムより大きいので、ECC スケジューリングをステップ 2 またはステップ 3 のどちらかで適用する必要がある。ステップ 3 で ECC スケジューリングを適用すると仮定する (記号 ‹ は ECC を適用するステップを表す)。同様に、d_2 にも ECC を適用する必要があるが、ECC モジュール数は 1 個なので d_2 にステップ 3 で ECC を適用することはできない。したがって、ステップ 2 と 4 の両方で ECC を適用しなければならない。しかし、ECC モジュール数の制約により d_3 に ECC を適用することはできない。この結果、ECC モジュールがもう一つ必要になる (合計 2 個)。d_1 に ECC を適用する際、ステップ 3 ではなくステップ 2 で適用すると、図 3(c) に示すように 1 つの ECC モジュールで ECC スケジューリングが可能である。さらに、ECC 適用回数の総数も減らすことができる ((b) では 4 回、(c) では 3 回)。この例題は ECC スケジューリングを最適化しながら、ECC モジュール数を最小化する問題の動機を与える。

3. 整数計画法としての定式化

整数計画法 (integer linear programming: ILP) は最適化手法として良く使われる。本章では、ECC スケジューリングの解空間を探索しながら ECC モジュール数を最小化する問題を ILP で定式化する。

3.1 問題の定式化

本稿で考える最適化問題を以下のように定式化する。演算スケジュール済の DFG、演算器の集合、レジスタの集合、ECC モジュール数の集合、セーフタイムが入力として与えられたとき、セーフタイム制約を各データについて満足し ECC モジュール数を最小化する ECC スケジューリングを出力する。

3.2 ILP 定式化

ILP の変数および定数の定義を表 1 に記載する。各データ d_i について ECC スケジューリングステップに仮の順序付けをする (すなわち、$j_1 < j_2$ ならば $x_{i,j_1} \leq x_{i,j_2}$)。各データ d_i について ECC スケジューリングステップはライフタイム $[L_i^{(s)}, L_i^{(e)}]$ の区間内になければならない。この制約は式 (1) で表現できる。

$$L_i^{(s)} \leq x_{i,1} \leq x_{i,2} \leq \cdots \leq x_{i,u(i)} \leq L_i^{(e)} \tag{1}$$

式 (1) において、変数 $x_{i,j}$ は同じ値を取り得る。したがって、一般的なケースを扱うことができる。

隣り合う ECC スケジューリングステップの間隔は X を越えてはならない。この制約は式 (2)–(4) で表現できる。

$$x_{i,1} - L_i^{(s)} \leq X \tag{2}$$

$$x_{i,k} - x_{i,k-1} \leq X, \quad 2 \leq \forall k \leq u(i) \tag{3}$$

$$L_i^{(e)} - x_{i,u(i)} \leq X \tag{4}$$

j 番目のステップで $y_{\max}$ は全てのステップにおいて ECC スケジューリングの最大数を下回ってはならない。この制約は式 (5) で表現できる。

$$\sum_{o_i \in O_j} x_{i,j} \leq y_{\max}, \quad 1 \leq j \leq S_{\max} \tag{5}$$

$y_{\max}$ の最小数は最小の ECC モジュール数に対応する。したがって、目的関数は以下の式で与えられる。

$$\text{Minimize} \quad y_{\max} \tag{6}$$

4. ケーススタディ

提案手法を高位合成のベンチマーク回路である 5 次楕円フィルタに適用した [14]。セーフタイムを 4 ($X = 4$)、ECC の適用に要するクロックサイクル数を 1 と仮定する。図 4(左) に示す演算スケジューリングは加算器 3 個、乗算器 1 個を仮定して設計した (全ての演算を 1 ステップ演算と仮定)。図 4(右) はライフタイムの集合を表す (ライフタイム長が 3 以下のライフタイムはセーフタイムより短いので ECC を適用する必要がないため表記を省略)。図 4(右) は単純な ECC スケジューリングの適用結果である。この結果、4 個の ECC モジュールを必要とし、ECC 適用の総数は 17 であった (17 はこの例題に対する最小数である)。

ⓒ 2016 Information Processing Society of Japan

表 1　ILP 定式化に用いる定数と変数の定義

記号	定義	種類
i, j	それぞれ、演算と ECC スケジューリングステップの番号を表す。	変数
$L_i^{(s)}, L_i^{(e)}$	d_i のライフタイムの開始と終了ステップ	定数
$x_{i,j}$	j^{th} i 番目の演算の出力に対する ECC スケジューリングステップ	変数
X	セーフタイム	定数
$u(i)$	d_i のライフタイム長	定数
$S_{\max}$	最大ステップ	定数
$y_{\max}$	ECC モジュール数	変数
O_j	出力データのライフタイムがステップ j を含む演算の集合	変数

一方、図 5(右) は提案手法による結果である。この場合必要な ECC モジュール数は 2 個であり、ECC 適用の総数は 17 であった。この結果から、提案手法は ECC モジュール数を削減するのに効果的であることが確認できた。本例題に関しては計算時間はほぼ 0 秒である。

5. 結論

本論文ではソフトエラー耐性を有する ASIC の高位合成問題について議論した。特にデータパスの記憶回路であるレジスタに着目し、ECC モジュールを付加したレジスタの最適化問題に取り組んだ。データのライフタイムの特徴に着目し、セーフタイムという新しい概念を導入した。ASIC の信頼性を維持しながら、ECC モジュール数の最小化問題を考え、整数計画法に基づく手法を提案した。例題によって提案手法の有効性を確認した。今後の課題としては提案した ILP 手法を ILP ソルバを用いて多くのベンチマーク回路に適用することや演算スケジューリングも考慮した最適化を行うことが挙げられる。

参考文献

[1] T. Uhrmann, T. Matthias, M. Wimplinger, J. Burggraf, D. Burgstaller,H. Wiesbauer, and P. Lindner, "Recent progress in thin wafer processing," *Proc. International 3D Systems Integration Conference (3DIC)*, pp. 1–8, Oct. 2013.

[2] K.-C. Wu and D. Marculescu, "Power-aware soft error hardening via selective voltage scaling," *Proc. International Conference on Computer Design (ICCD)*, pp. 301–306, Oct. 2008.

[3] M. Maniatakos, M.K. Michael, and Y. Makris, "Vulnerability-based interleaving for Multi-Bit Upset (MBU) protection in modern microprocessors," *Proc. International Test Conference (ITC)*, pp. 1–8, Nov. 2012.

[4] S. Ghosh, S. Basu, and N.A. Touba, "Reducing power consumption in memory ECC checkers," *Proc. International Test Conference (ITC)*, pp. 1322–1331, Oct. 2004.

[5] S. Tosun, N. Mansouri, E. Arvas, M. Kandemir, and Y. Xie, "Reliability-centric high-level synthesis," *Proc. Design, Automation and Test in Europe (DATE)*, pp. 1258–1263, vol. 2, Mar. 2005.

[6] L. Chen, M. Ebrahimi, and M.B. Tahoori, "Reliability-aware operation chaining in high level synthesis," *Proc. European Test Symposium (ETS)*, pp. 1–6, May 2015.

[7] W.R. McKee, et. al, "Cosmic ray neutron induced upsets as a major contributor to the soft error rate of current and future generation DRAMs," *Proc. International Reliable Physics Symposium (IRPS)*, ppp. 1–6, 1996.

[8] V. Degalahal, R. Ramanarayanan, N. Vijaykrishnan, Y. Xie, and M.J. Irwin, "The effect of threshold voltages on the soft error rate," *Proc. International Symposium on Quality Electronic Design (ISQED)*, pp. 503–508, Mar. 2004.

[9] J. Oh and M. Kaneko, "Automated selection of check variables for area-efficient soft-error tolerant datapath synthesis," *Proc. International Symposium on Circuits and Systems (ISCAS)*, pp. 49–52, May 2015.

[10] S.Z. Shazli and M.B. Tahoori, "Soft error rate computation in early design stages using boolean satisfiability," *Proc. Great Lakes symposium on VLSI*, pp. 101–104, May 2009.

[11] T. Iwagaki, Y. Ishimori, H. Ichihara, and T. Inoue, "Designing area-efficient controllers for multi-cycle transient fault tolerant systems," *Proc. European Test Symposium*, May 2015.

[12] G. Rui, C. Wei, L. Fang, D. Kui, and W. Zhiying, "Modified triple modular redundancy structure based on asynchronous circuit technique," *Proc. International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 184–196, Oct. 2006.

[13] T. Heijmen and A. Nieuwland, "Soft-error rate testing of deep-submicron integrated circuits," *European Test Symposium (ETS)*, pp. 247–252, May 2006.

[14] P. Michel, U. Lauther, and P. Duzy, *The Synthesis Approach to Digital System Design*, Kluwer Academic Publishers, 1992.

ⓒ 2016 Information Processing Society of Japan

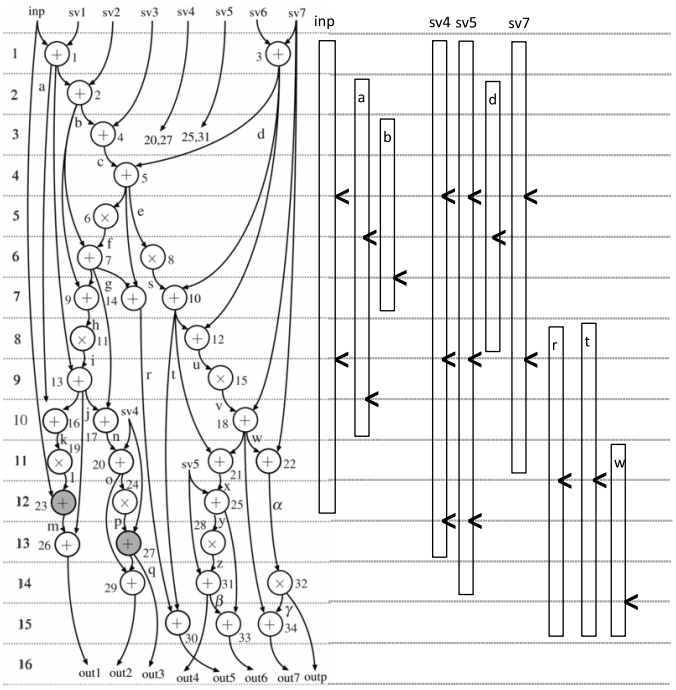

図 4 EWF5 scheduling result (left) and the lifetimes (right). Note that in this example, safe-time X is 4, therefore the short-time lifetimes (less than 5 steps) are not shown. The right figure shows a naive ECC scheduling resullt without considering minimizing the number of ECC modules. The number of applying ECC is 17, and the number of ECC modules is 4.

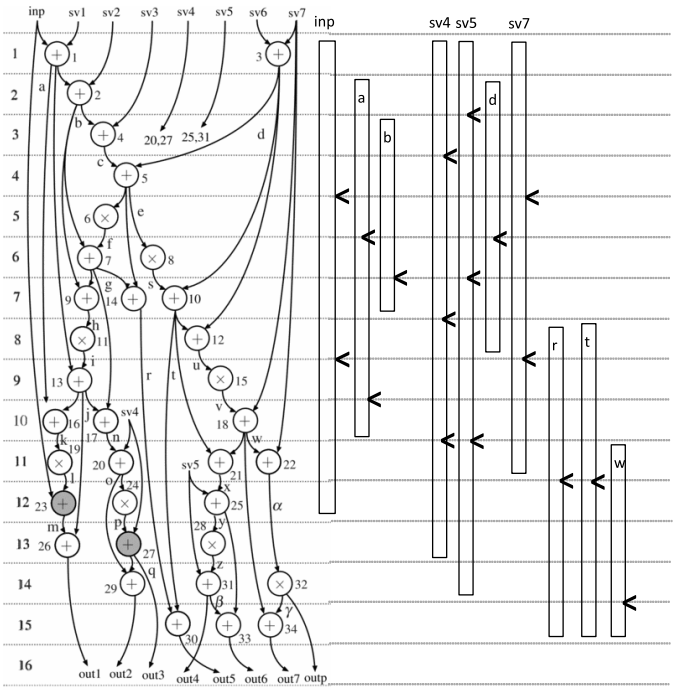

図 5 The proposed ECC scheduling result with considering minimizing the number of ECC modules. The number of applying ECC is 17, and the number of ECC modules is only 2.

ⓒ 2016 Information Processing Society of Japan