科学研究費助成事業　　研究成果報告書

平成　２６　年　　６　月　１６　日現在

機関番号：　２５４０３
研究種目：　基盤研究(C)
研究期間：　2011～2013
課題番号：　２３５０００６５

研究課題名（和文）マルチサイクル過渡故障に耐性を持つディジタルシステムの動作合成法

研究課題名（英文）A method of high-level synthesis for multi-cycle transient fault tolerant digital systems

研究代表者
井上　智生（INOUE, Tomoo）

広島市立大学・情報科学研究科・教授

研究者番号：４０２５２８２９

交付決定額（研究期間全体）：（直接経費）　　2,900,000円、（間接経費）　　870,000円

研究成果の概要（和文）：大規模集積回路（LSI）の微細化により，ソフトエラー（放射線衝突による一時的な誤り）の発生が無視できなくなっている．本研究では，LSI システムに発生するソフトエラーのような過渡故障に耐性を持つ（内部の誤りを外に出さない）システムの合成法を提案する．
提案法は，動作記述を入力として，一定時間継続する過渡故障に耐性を持つレジスタ転送レベル回路を出力する．出力される回路は機能的に3重化されている．データパス部では3つのユニット間で適切に演算器を共有し，さらに，その過渡故障耐性を利用してコントローラ部も耐過渡故障化させる．従来の一般的な耐故障設計法に比べてより小さい面積で実現可能である．

研究成果の概要（英文）：As the advance in semiconductor technologies, transient faults caused by particle strike have become a matter of concern, and further it is predicted that such faults can span across more than one clock cycle.
We proposes a high-level synthesis algorithm for long duration transient fault tolerance. On the basis of the properties of operational units for transient error correction and detection among operations, we present (1) a force-directed scheduling algorithm with the force derived from the estimation of operational units according to the properties of error correction and detection, (2) a binding algorithm aiming at achieving both of correctablility and detectability with minimizing the number of operational units, and (3) a method for synthesizing transient fault tolerant controllers that utilizes the correctability/detectability of datapaths. The proposed algorithm can derive multi-cycle fault tolerant systems with small hardware resources compared with simply-tripled datapaths.

研究分野：　総合領域

科研費の分科・細目：　情報学・計算機システム・ネットワーク

キーワード：　ソフトエラー　高位合成/動作合成　過渡故障　信頼性　3重系/TMR　誤り訂正・誤り検出　耐故障設計

1　研究開始当初の背景

大規模集積回路 (LSI) の用途が広がるにつれてその信頼性の実現，確保が大きな課題となっている．LSI の信頼性を阻害する要因には，物理的な欠陥（ハードウェア故障）の他にソフトエラー（放射線衝突による一時的な誤り）がある．近年の微細化に伴い，動作中の LSI に発生するソフトエラーが無視できない状況にあり，更に放射線衝突の影響は単一サイクルに収まらず複数のクロックサイクルにまたがる可能性が懸念されている．動作中の障害は大きな社会的損害につながる可能性もあり，したがって，オンライン（システムが動作している状態）での信頼性の確保のための技術が必要不可欠となっている．

一方，これまでの高信頼化設計の一般的なアプローチに TMR (triple module redundancy: 三重系) がある．TMR は 1 つのユニットで発生する誤りを訂正できるが，単純な 3 重化は面積増加を招き，物理的欠陥やソフトエラーなどの一時故障の発生確率を増加させ，よって必ずしも高い信頼性を実現できるとは限らない．したがって，従来の面積・遅延だけでなく，信頼性も含めた最適化設計を指向する手法が必要となる．

2　研究の目的

本研究では，LSI のオンライン（システムが動作している状態）での高信頼化に重点を置いた動作合成法の提案を目的とする．従来の物理的欠陥の存在・発生を回避することを主目的としたものとは異なり，提案する設計法は LSI システムが動作中に発生する外的要因（放射線衝突）による一時的な誤り（過渡故障）を主眼を置くものである．発生しうる過渡故障の性質に応じた必要かつ十分な冗長性を実現することで，過剰な耐故障設計を避け，高信頼システムを低コストで実現可能にするものである．今日あるいは将来の情報化社会の基盤となる LSI にとって，本研究の成果は学術的な特色を持つだけではなく，半導体業界や社会にも大きな貢献をもたらすものと期待される．

3　研究の方法

(1) 本研究で提案する動作合成アルゴリズムの出力となるレジスタ転送レベル回路のモデル化，および，そのレジスタ転送レベル回路が持つべき耐過渡故障性（検出可能／訂正可能な故障期間）に関する条件を整理する．

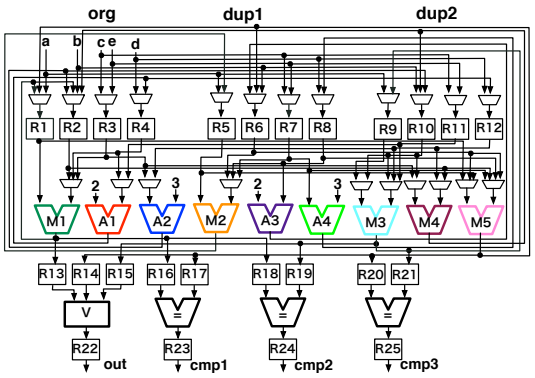

図 1: 2 サイクル誤り訂正／4 サイクル検出可能な 3 重化データパス ($k_c = 2, k_d = 4$)

(2) 上述のモデル化と条件に基づくデータパス部の合成法を提案する．データパス部の合成を，スケジューリング，バインディングのステップに分けて考察する．スケジュール済みデータフローグラフに対して，過渡故障の訂正条件，検出条件を満たすバインディングにおける下界を示し，その下界を利用したヒューリスティックアルゴリズムを考察する．

さらに，スケジュール済みデータフローグラフに対する演算器数の下界の算出方法を応用して，未スケジュールデータフローグラフに対する演算器数を予測する方法を考察し，それを用いたスケジューリングアルゴリズムを提案する．

(3) 耐過渡故障性を持つデータパス部に対する制御信号を生成する耐過渡故障コントローラ部の合成法を考察する．全体として，要求される耐過渡故障性に対して最適なレジスタ転送レベル回路を合成可能とする．

4　研究成果

4.1　耐過渡故障レジスタ転送レベルデータパス

誤り継続期間が複数サイクルにおよぶソフトエラーは過渡故障としてモデル化できる．故障期間が k サイクル継続する故障を k サイクル故障と呼ぶ．以下では，過渡故障は演算器に発生するものとし，故障による誤りが生じる期間内では複数の演算器では発生しないものとする．

本研究では，耐マルチサイクルエラーを指向した高位合成法の対象として，機能的に 3 重化されたデータパスを考える．図 1 に耐マルチサイクルエラーを指向したデータパスの例と，図 2 に図 1 のデータフローグラフ (DFG) を示す．図 2 の各頂点は演算を，M は演算に割り当てられ

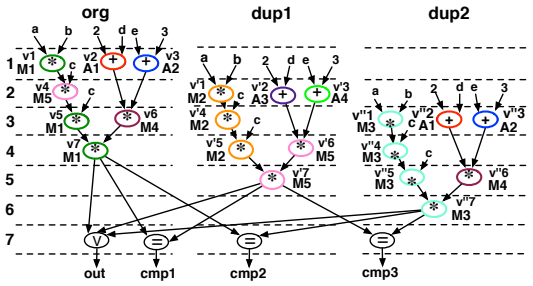

図 2: 図 1 の動作記述

ている演算器を示す．org. は主演算部（オリジナルのユニット）であり，副演算部である dup.1, dup.2 はいずれも org. と同じ動作をする．3 つのユニットの出力は多数決回路によって誤り訂正され，比較器によって誤り検出される．耐マルチサイクル過渡故障システムは，k_c サイクル誤り訂正／k_d サイクル誤り検出の能力を持つものとする．

すなわちこのシステムでは，誤りが存在するユニットが 1 つまでならば多数決回路を通して訂正でき，2 つまでならばいずれかの比較器で検出可能である．

与えられたデータフローグラフからこのような誤り訂正能力／誤り検出能力を有するデータパスを合成するための，演算器の共有とそのスケジュール (各演算の開始時刻) に関する条件は次のように整理できる．

演算器共有の条件 各演算器に割り当てられた任意の演算 v_i, v_j について次のすべてが成り立つ．

一般共有可能条件: 演算の実行時刻に重複がない．

誤り訂正条件: 異なるユニットでは 2 つの演算 v_i, v_j 間の時刻差 (|(v_iの開始時刻) − (v_jの開始時刻) + (演算 v_jの遅延)|) が k_c サイクル以上である．

誤り検出条件: 演算 v_i, v_j が属するユニットが異なり，かつ，同じ演算器に割当てられている第 3 の演算 v_k が v_i, v_j と異なるユニットに存在するならば，その実行時刻は v_i, v_j のいずれかから k_d サイクル以上の差がある．

4.2 バインディングアルゴリズム

演算器バインディングでは，上述の演算器の共有条件より，システムに必要な演算器数の下界を訂正／検出条件ごとに見積もる．これは，訂正条件と検出条件によって必要となる演算器数の下界が一般共有可能条件のそれと異なるからである．提案するヒューリスティックはこの下界と，ユニットの共有均衡度を尺度とする．アルゴリズムの入力はスケジュール済み DFG(SDFG), 誤り訂正可能サイクル数 k_c，誤り検出可能サイクル数 k_d，出力は演算器バインディングの結果，最適化目標は演算器数最小である．

訂正条件によって求められる演算器数の下界を時刻ごとに見積もる．訂正条件より，時刻 t から k_c サイクル誤りが発生した場合にユニット u で必要となる演算器 τ の数は，期間 $[t, t+k_c-1]$ 内に存在する演算数の最大数であり，各ユニットの値の合計が時刻 t における演算器数の下界である (式 (1))．図 2 において k_c=2，時刻 3 の乗算の下界は $2+2+1=5$ となる．検出条件によって求められる演算器数の下界も時刻ごとに見積もる．訂正条件と検出条件の異なる点は，誤りが継続する期間が k_d サイクルであり，その期間内ならば，2 つのユニットで同じ 1 つの演算器を共有できる．その 2 ユニット間での共有が均衡すれば (すなわち，{org., dup.1}, {org., dup.2}, {dup.1, dup.2} のそれぞれの対での共有数が同じになれば)，演算器数は演算数の 1/2 となり，これを下界とできる．しかし，検出条件の下界は一般共有可能条件から求まる下界より低く見積もる可能性があるので，このときは一般共有可能条件の下界を時刻 t の下界とする (式 (2))．図 2 において k_d=4，時刻 3 の乗算の下界は $(2+2+2)/2=3$ となる．訂正条件と検出条件の時刻ごとの下界を $1 \le t \le L$ について求め，最大の値がシステムの下界となる (式 (3))．$|V_l(t', \tau, u)|$ はユニット u における一般共有可能条件より求まる演算数，U は {org., dup.1, dup.2} を表す．

$$n_c(t, \tau, k_c) = \sum_{u \in U} \max_{t \le t' < t+k_c-1} |V_l(t', \tau, u)| \quad (1)$$

$$n_d(t, \tau, k_d) = \max\{\lceil \frac{n_c(t, \tau, k_d)}{2} \rceil, |V_l(t, \tau, u)|\} \quad (2)$$

$$n_L(\tau, L, k_c, k_d) = \max\{ \max_{1 \le t \le L} n_c(t, \tau, k_c), \max_{1 \le t \le L} n_d(t, \tau, k_d)\} \quad (3)$$

共有の均衡について例を用いて説明する．図 2 において $k_d = 3$ として加算器に着目する．演算器を $\{v_2, v_2''\}, \{v_3, v_3''\}$ のように共有すると v_2' と v_3' は共有できないので共有の均衡がとれず演算器数は 4 となる．一方で，$\{v_2, v_2''\}, \{v_2', v_3''\}$ のような共有ならば $\{v_2'', v_3'\}$ が共有ができ，共有の均衡がとれるので演算器数は 3 となる．

提案するヒューリスティックでは初めに各時刻の下界を求め，バインディング途中では期間内に割当てられている演算器数と下界が大きい期

間から割り当てを行う．共有する演算対は共有可能性が高く，共有するユニットが偏らないように共有する．

4.3 スケジューリングアルゴリズム

スケジューリングでは，上述したスケジュール済みデータパスに対する演算器数の下界を予測しながら，その値を小さくすることを目標とする．提案するアルゴリズムの入力は DFG，k_c，k_d, レイテンシ制約 L，出力は SDFG であり，フォースディレクテッドスケジューリング (FDS) に基づいたスケジューリングである．

FDS は，演算同士がバネでつながれて互いに力 (フォース) が作用すると見なし，互いの負荷が最小となる状態が最適なスケジュールと考えるアルゴリズムである．具体的には，演算 v_i を時刻 t にスケジュールしたときのフォース $f(v_i, t)$ を求め，最も小さいフォースになる演算から順にスケジュールすることで，全体の演算器数の最小化を指向する．フォースは，演算 v_i が各時刻 t に存在する確率のスケジュール前後の変化と各時刻 t において必要な演算器数から求められる．

提案法ではバインディングと同様に訂正／検出条件によって存在確率の求め方が異なる．フォースは訂正／検出の各条件によって求められた存在確率の変化を加えることにより求められる．誤り訂正条件より，演算 v_i が存在確率をもつ各時刻 t において，演算 v_i が属するユニットでは同時刻に実行される演算から，他のユニットでは時刻差が k_c サイクル以下にある演算からフォースを受ける (式 (4))．時刻 t でユニット u の演算が受ける力は，時刻 t で必要な演算器数である．誤り検出条件より，演算 v_i が存在確率をもつ各時刻 t において，誤り訂正条件と同じく k_d サイクル以内の演算からフォースを受けるが，検出条件では 2 つのユニットまでなら共有が可能なので受けるフォースは 1/2 となると考える．

バインディング時と同様に，一般共有条件のフォースのほうが大きければこれを受けるフォースとする (式 (5))．これらは，演算 v_i がどのユニットに属するかによって受けるフォースが異なる．$|V(t,\tau)|$ は一般共有条件より求まる演算数を表す．時刻ごとの下界を $1 \le t \le L$ について求め，最大の値がシステムに必要な演算器数となる (式 (6))．

$$n_{c\_step}(t, k_c, \tau, u) = \mid V_l(t, \tau, u) \mid + \max_{t-k_c+1 \le t' \le t} \sum_{\omega \in U \mid \omega \neq u} \max_{t' \le t'' \le t'+k_c-1} \mid V_l(t'', \tau, \omega) \mid \quad (4)$$

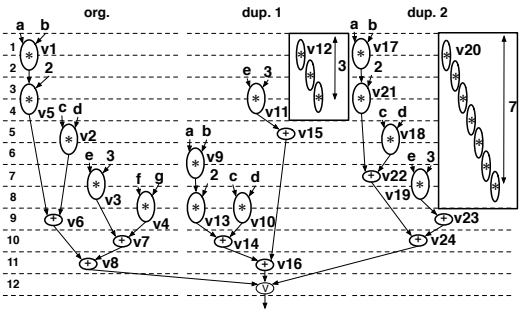

図 3: 未スケジュールの演算が存在する DFG

$$n_{d\_step}(t, k_d, \tau, u) = \max\{ \frac{n_{c\_step}(t, k_d, \tau, u)}{2}, |V(t,\tau)| \} \quad (5)$$

$$n_{L\_step}(\tau, L, k_c\, k_d) = \max\left\{ \max_{1 \le t \le L} n_{c\_step}(t, \tau, k_c), \max_{1 \le t \le L} n_{d\_step}(t, \tau, k_d) \right\} \quad (6)$$

ヒューリスティック尺度として，与えられた DFG において，スケジュール候補の演算に対して，スケジュール可能な時刻 t へ割り当てたときのフォースを考える．演算 v_i がスケジュールされることでフォースが作用するのは，演算のスケジュール確率が変化した演算であり，演算 v_i が属するユニットである．

図 3 において $k_c = 2, k_d = 4$ のとき，演算 $v20$ を時刻 5 にスケジュールしたときのフォースを考える．演算 $v20$ の存在確率は，時刻 1 は $1/7 \rightarrow 0$，時刻 5 は $2/7 \rightarrow 1$ に各時刻で変化する．各時刻で dup.2 において同時刻にある演算から訂正条件により受けるフォースは，ユニット dup.2 の時刻 t で必要な演算器数であるので $\widetilde{n}_{c\_step}(t, 3, \text{乗算}, \text{dup.2})$ である．フォースは，$f(v_{20}, 5, 2) = (-1/7)\widetilde{n}_{c\_step}(1, 2, \text{乗算}, \text{dup.2}) + (-2/7)\widetilde{n}_{c\_step}(2, 2, \text{乗算}, \text{dup.2}) + \ldots + (-1/7)\widetilde{n}_{c\_step}(7, 2, \text{乗算}, \text{dup.2})$ と表される．同様に検出条件により受けるフォースは $\widetilde{n}_{d\_step}(t, 4, \text{乗算}, \text{dup.2})$ であり，フォースも k_c を k_d に置き換えることにより同様に求められる．全体のフォースは訂正条件のフォースと検出条件のフォースを足した値となる．同じように全ての未スケジューリングの演算 v を各時刻 t へスケジュールしたときのフォースを求め，フォースが最小となるスケジュールを選択する．

表 1 に，上述のスケジューリングアルゴリズムとバインディングアルゴリズムを適用した k_c サイクル誤り訂正，k_d サイクル誤り検出可能データパスにおける演算器数を示す．この表より，ほとんどの場合単純な 3 重化より演算器数を削減することができ効果を確認できた．

表 1: 提案法における各 k_c, k_d に対する演算器数

DFG	リソース	単純な 3 重化	提案法 (k_c, k_d)		
			(1, 1)	(2, 5)	(3, 7)
JWF (L=13)	加算器	6	3	3	3
	乗算器	3	4	4	4
EWF (L=21)	加算器	9	7	6	7
	乗算器	6	4	5	6

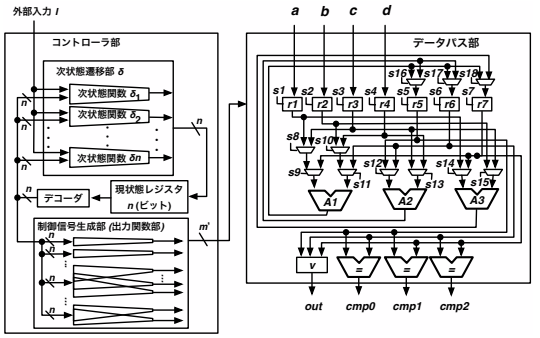

図 4: 耐マルチサイクル過渡故障システムのコントローラ部とデータパス部

4.4 コントローラ

提案するコントローラは，主に次状態関数部と制御信号生成部からなる．次状態関数部は，単一故障に耐性を持つ SID (Single Independent Decoder) を利用する．制御信号生成部は後述する設計法によって，過剰な耐故障化を避けた小面積な耐過渡故障コントローラを実現する．

制御信号は被制御要素 (演算器，MUX, レジスタ) と一対一に対応する．制御信号の誤りは，制御対象のデータパス部の構成要素の誤りと同様に考えることができる．そのため，コントローラの制御信号生成部で k サイクル故障が生じても，データパス部の誤り訂正／検出能力によって訂正できる場合がある．図 4,5 の例をもとに説明する．制御信号 s_{14} と s_{18} において同時に時刻 4 から 5 まで誤りが生じたとする．s_{18} はレジスタ r_7 の入力を選択する MUX を制御している．r_{18} はユニット dup.2 に属するので誤りの影響は dup.2 に及ぶ．また s_{14} は演算器 A_3 の入力を切り替える MUX を制御している．A_3 は時刻 4 において演算 v_7 を行うので，誤りの影響はユニット dup.2 に及ぶ．A_3 は時刻 5 において演算 v_8 を行うので，誤りの影響はユニット dup.2 に及ぶ．よって時刻 4 から 5 までの間に影響を及ぼすユニットは dup.2 だけなので，これはデータパス部の 2 サイクル誤りに対する訂正能力をそのまま利用して訂正ができる．したがって，制御信号生成部は，制御信号間で一部の論理を共有できる可能性があり，全体として回路を簡単化できる可能性がある．

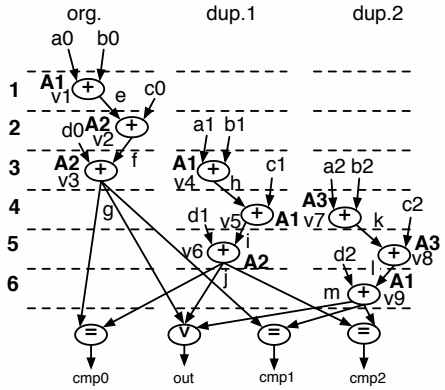

図 5: 図 4 の動作と演算器バインディング

制御信号生成部を構成する独立部分回路の数が少ないことが全体として論理回路規模を最小化できると仮定し，以下の問題を定式化する．

制御信号集合分割問題

入力：制御信号集合 $S = \{s_1, s_2, \ldots, s_m\}$, 誤り訂正サイクル数 k_c, 誤り検出サイクル数 k_d

出力：制御信号集合の分割 $\pi = \{S_1, S_2, \ldots, S_{m'}\}$

最適化目標：分割数 m'（$\leq m$）の最小化

この分割 π の各ブロック S_i は，対応する制御信号生成回路の一部の論理が共有可能であることを意味する．例えば図 4 において，$(k_c, k_d) = (2, 5)$ のとき，外部入力レジスタの制御を除く制御信号集合を，

$$S = \{s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}, s_{12}, s_{13}, s_{14}, s_{15}, s_{16}, s_{17}, s_{18}\}$$

とすると，この制御信号集合の最適分割は次のようになる．

$$\pi = \{\{s_5\}, \{s_9, s_{11}\}, \{s_{12}, s_{13}, s_{16}\} \{s_6, s_8, s_{10}, s_{17}\}, \{s_7, s_{14}, s_{15}, s_{18}\}\} \quad (7)$$

すなわち，制御信号生成部は 5 つの部分回路で構成できる．

データパス部の合成結果から制御信号抽出して上述の問題を解くプログラムを実装し実験を行った．得られた結果から Design Compiler™ (Synopsys 社) の合成を行った．セルライブラリには NanGate 45nm を用いた．結果の一部を表 2 に示す．効果的に面積を削減できることがわかる．

表 2: 実験結果（コントローラの面積）

SDFG	k_c	k_d	単純 3 重化	SID+法	提案法
jwf1	2	5	361.13	253.02	207.99
	3	7	310.81	227.45	184.54
jwf2	2	5	267.69	183.75	158.55
	3	7	258.61	184.20	161.62

5　主な発表論文等

〔雑誌論文〕（計 2 件）

(1) Tsuyoshi Iwagaki, Tatsuya Nakaso, Ryoko Ohkubo, Hideyuki Ichihara, Tomoo Inoue, ”A Heuristic Algorithm for Operational Unit Binding to Synthesize Multi-Cycle Transient Fault Tolerant Datapaths,” Digest of Papers 14th IEEE Workshop on RTL and High Level Testing (WRTLT), Nov. 2013.

(2) Tomoo Inoue, Hayato Henmi, Yuki Yoshikawa, Hideyuki Ichihara, ”High-Level Synthesis for Multi-Cycle Transient Fault Tolerant Datapaths,” Proc. IEEE Int. On-Line Testing Symp. (IOLTS), pp. 13-18, July 2011.

〔学会発表〕（計 2 件）

(1) 石森裕太郎, 中祖達也, 岩垣剛, 市原英行, 井上智生, ”耐マルチサイクル過渡故障を指向した高位合成におけるコントローラの設計について,” 信学技報 (DC2013-34), Vol. 113, No. 321, pp. 45-50, 2013 年 11 月.

(2) 中祖達也, 大窪凉子, 岩垣剛, 市原英行, 井上智生, ”耐過渡故障データパス合成における演算器バインディングのためのヒューリスティックアルゴリズム,” 信学技報 (DC2012-50), Vol. 112, No. 321, pp. 147-152, 2012 年 11 月.

〔その他〕ホームページ
http://www.cd.info.hiroshima-cu.ac.jp/~tomoo/

6　研究組織

(1) 研究代表者
井上　智生 (INOUE, Tomoo)
広島市立大学・大学院情報科学研究科・教授
研究者番号：40252829