論文の内容の要旨

論文題目　　システムＬＳＩの上流設計自動化に関する研究

氏　　名　　若林　一敏

本論文は、システムＬＳＩの上流の設計工程である機能設計工程を自動化するＣ言語ベースの統合設計環境と、その中核をなす動作合成手法について論じる。ＬＳＩはその誕生以来、年々大規模化、複雑化を進め、それらを搭載する電子装置の小型化、低コスト化に大きな役割を果たしてきている。近年では、システム全体が１つのチップに搭載する「システムＬＳＩ」が普通に使われる程大規模化が進んでいる。このような、製造技術の進歩によるＬＳＩの大規模化、複雑化のペースに設計生産性の向上が追いつかず、より上流の設計工程の自動化による設計生産性の向上が必須である。80 年代前半から下流工程から順に自動化がすすみ、レイアウト設計、論理設計の自動化がすでに実用化されている。本論文はこのさらに上流の機能設計の自動化手法を提案する。

最初に、機能設計の自動化を行うことの意義を述べる。現在主流のＲＴＬ設計に比べ動作レベル設計では記述量が数分の一になること、シミュレーションが数十倍から数百倍になることを述べる。また、提案システムが動作記述言語としてＣ言語を採用した理由を述べる。システムＬＳＩの組み込みソフトウェアや、ハードウェア化される信号処理アルゴリズム等が主にＣ言語で書かれているため、ソフトウェア設計者、ハードウェアのアルゴリズム設計者と、ＬＳＩ設計者が共通の言語で協調設計できる利点がある。

次に提案するＣ言語ベースのシステムＬＳＩの設計環境（図 1）について論じる。本環境は、Ｃ言語を入力としＲＴＬを自動合成する動作合成ツール Cyber と、形式検証、ハードウェアソフトウェア協調検証、フロアプラン、解析ツール等からなり、すでに多数の実チップを設計した実績がある。本環境の基本コンセプトは”All-in-C”設計である。即ち、制御系回路もデータ処理系回路も、システムＬＳＩのすべての回路をＣ言語で記述し合成できること。また、設計も検証もシステムＬＳＩの上流設計工程を基本的にＣ言語だけで行えること（つまり、合成したＲＴＬをデバッグする必要の無いこと）の 2 点である。このコンセプトの実現のために、各ツールに以下の項目を実現するための機能を研究し、ツールを統合化した。まず、提案する動作合成

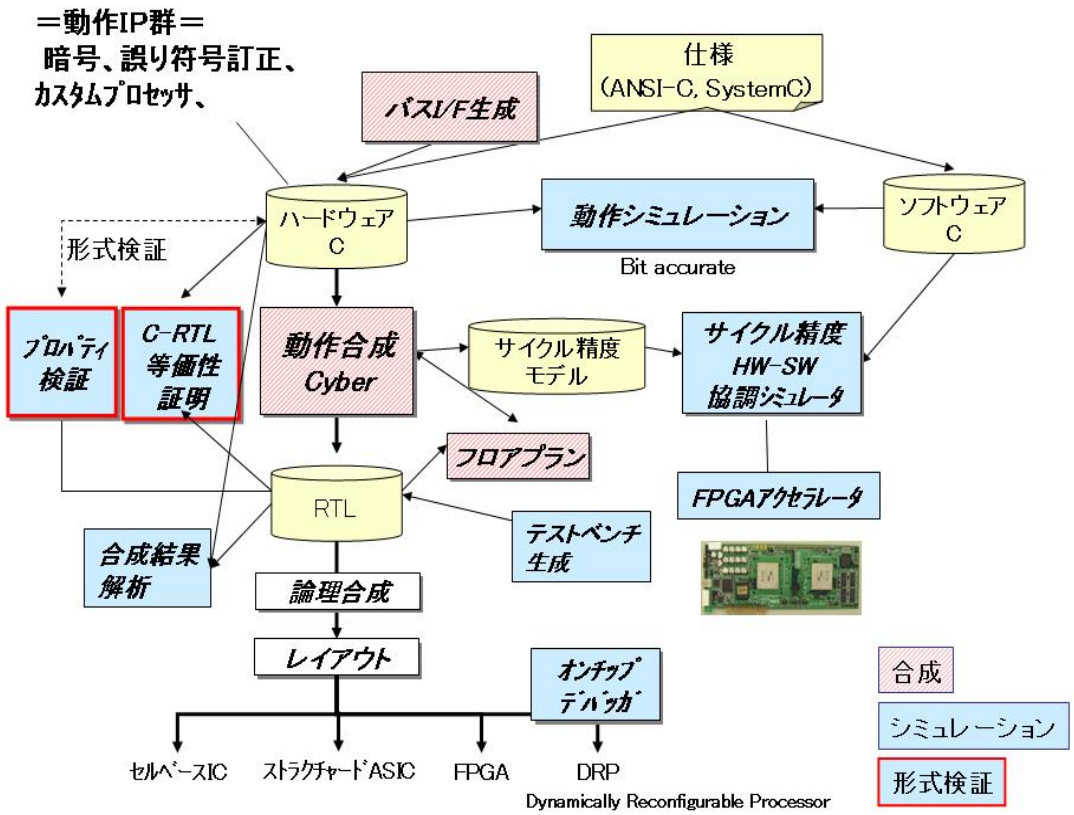

図1. C言語ベースのシステムLSIの統合設計環境の構成図

手法は、制御系回路、データパス系回路の双方を十分な性能で合成するための複数の合成機能を持たせた。特に、複雑な制御構文を持つ動作記述からでも高性能な回路を合成する手法を提案した。また、設計者の与える回路特性（プロパティ）を合成されたＲＴＬ記述が満たすことを証明する「プロパティ検証ツール」で、プロパティをＣソースでの変数名を用いて記述できるようにした。「ＣとＲＴＬ等価性証明ツール」を開発することで動作合成ツールのバグによる回路の不正合成を未然に防ぎ、ＲＴＬシミュレーションの必要性を減じた。さらに、「ハードウェアとソフトウェアの協調シミュレータ」は、その実行、デバッグを合成されたＲＴＬ上でなく、Ｃソースコード上でできるようにした。また、ＲＴＬシミュレーションに比べ高速なシミュレーションを可能にするサイクル精度のＣモデルを生成する技術も提案した。

本Ｃベース設計環境は大きなシステムであり、筆者をリーダとする研究グループ全体の成果である。筆者は全体システムの中の核となる動作合成ツールの研究開発を行うとともに、本Ｃベース設計環境の全体構想（提唱した「all-in-C コンセプト」を実現するためのモジュール構成と機能、基本手法）を提案し研究してきた。従って、本論文では動作合成手法の詳細と、合成結果を中心に論じ、Ｃベース設計環境全体の有効性については、動作合成の視

点から考察を加える。

動作合成ツールは、ソフトウェアのＣコンパイラに比べ、並列化性能がより重要である。ハードウェアはソフトウェア（ＣＰＵ）に比べ、搭載する演算器、独立にアクセス可能なメモリ、配線等のハードウェア資源を豊富に、かつ自由な構成で利用することで、ＣＰＵよりも高性能、低面積、低電力な回路でなければならないからである。ソフトウェアコンパイラの演算レベルの並列化を除外する大きな問題は、条件分岐等の制御依存性である。近年は、H.264 等のデータ動画圧縮、ビタビデコーダ等の誤り訂正等従来のハードウェアアルゴリズムに比べ、より複雑な分岐制御を行う複雑なアルゴリズムが主流になってきている。従って、より高い並列性を抽出するために制御依存を超えて並列化できる手法が必要である。本論文では、この制御依存を超えた並列化技法を提案し、ほとんどの投機実行を可能とする大局的スケジューリング手法を提案した。本手法は、条件ベクタ（Condition Vector）と呼ばれる条件排他性の表現形式と、これを利用したスケジューリング、バインディング、制御生成手法からなる。従来、VLIW 向けコンパイラや他の動作合成向けスケジューリング手法で提案されていた大局的スケジューリング手法群は、１つ１つの命令コードを条件節の上下に移動（命令移動:Code Motion）することで、いくつかの形式の投機実行を利用した並列化を行っていた。提案手法は、実際に命令を移動することなく、広範な種類の投機実行による並列化を、効率的な計算量（投機実行が無いときと同等）にて実行できる。

条件ベクタ（ＣＶ）は、ネストした条件分岐の全リーフ数（分岐節＋1）の次元を持つベクタ表現であり、各リーフの条件にそれぞれワンホットエンコードを与え、その親の分岐のＣＶは子の持つＣＶの論理ＯＲとする。ＣＶはＣ記述内の演算や配列アクセス、入出力等に与えられ、動作合成ツール内では CDFG の演算ノードに持たせる。ＣＶは、演算ノード同士の条件排他性を簡単にテストできる（同一位置に１が無いこと）、同一状態で必要になる演算器数が簡単に計算できる（全ＣＶのベクタ和を取り、最大値が必要演算器数）等の特長があり、分岐考慮による計算効率劣化が無い。さらに重要なのは、スケジューリング時に考えうるほとんどの投機実行を行った場合を、ＣＶの「動的排他性」という概念で表現可能なことである。特に、ネストした条件節の内側の一部の条件節だけの投機実行、トップレベルのＩＦ文の並列化等、従来法では考慮外としていたされていた並列化もスケジューリング時に計算量を増加することなく探索できる。本手法は、これらの特長以外に、最長パスの最適化だけでなく、全分岐パス群を最適化できるという長所もある。ＣＶは、バインディングや状態遷移生成でも利用されており、提案する

動作合成手法全体の基礎となっている。実験結果により本手法が制御依存を超えて並列性を十分に抽出できることを示す。

高い並列化を得るためには、制御依存性の扱いの他に、配列とループ、関数の取り扱いが重要である。従来の動作合成ツールでは、これらを合成したい回路形式を意識したＣ記述に書き換えることによって、高性能化しなければならないものがあったが、本ツールでは、これらに対応する様々な回路形式の知識を合成アルゴリズムの中に導入し、性能向上のため、低面積化のため等目的にあった回路を自動合成できるようにした。他にも、実製品に適用できるための性能の回路を合成するためには、数多くの最適化機能、合成制御機能が必要であり、本論文では作成したこれらの機能を整理分類して、議論する。

次に、ハードウェア記述言語と詳細なタイミング仕様をもつ回路向け合成手法について議論する。システムＬＳＩの内部には、ＤＭＡコントローラやバスブリッジ、液晶コントローラなどのように、仕様が入出力信号のタイミングチャートで与えられるようなモジュールも多々存在する。これらの回路は、先に示したような自動スケジューリングにタイミング制約を与えるよりも、クロックを動作記述に直接記述した方が直感的であり、記述効率が良い。また、プロセッサ設計時等で必要な割り込み動作の記述能力が汎用のＣ言語にはない。これらに対応するために、Ｃ言語へ加える新しい拡張表現（文法）を考案し、その拡張表現を生かした制御回路向け合成手法を提案した。

次の章では、提案した動作合成ツールを用いて合成した回路を解析し、簡単な統計処理により、ＲＬＴ記述に比べた動作記述量の削減効果が 1／7 程度あったことや、動作合成ツールが作成するサイクル精度シミュレーションモデルのＲＴＬシミュレーションに比べ数十倍から数百倍高速なことを示す。また、動作合成された回路品質の例として、カスタムプロセッサの合成結果を検証し、先に述べた記述量削減やシミュレーション速度向上の恩恵を享受しつつ、人手並みの回路品質で合成できたことを示す。次に、システムＬＳＩの設計期間・設計工数の削減、面積や電力の削減、動作記述の再利用性の高さ（生産性向上）、信頼性の向上（バグの削減）などを、提案ツールで合成した実例をもって分析し、効果を検証した。さらに、Ｃ言語設計に対するよくある批判に対して、本動作合成システムで合成した事例を用いて考察を行う。たとえば、目標遅延に達成するタイミングクロージャがＲＴＬ設計の場合よりも動作合成を利用したＣ設計のほうが良くなるケースの理由を述べる。次に、人手のＲＴＬ設計と動作設計について、そのハードウェア設計の発想法そのものの違い、実現されるハードウェア構成（回路形式）の違いについて考察し、動作合成の方が人手ＲＴＬよりも、優れた回路を出す

場合の理由についても実例を用いて議論する。特に、従来のＲＴＬ設計ではとうてい実現不能であった回路形式や、従来ＲＴＬ設計では非現実的である新しいハードウェアとソフトウェアの分割方法を提案し、実例を用いてその効果を説明する。最後に、今後の課題と結論を述べる。