

HDL を用いた LCA 開発手法

Design method for LCA Using HDL

船山 智* 丹生 和男* 黒沢 秀行**
Funayama, Satoshi Tansho, Kazuo Kurosawa, Hideyuki

FPGA's have become increasingly popular for fast prototyping and a small scale production. In order to optimizing LCA design, reducing design cycle time and re-targeting from LCA to gate arrays, we have built the integrated design environment using Hardware Description Language. We implemented the data conversion tool system between LCA and gate arrays, which contained equivalent symbols of gate array, look up table and data conversion programs. Using this integrated design environment, the design cycle time of LCA was reduced more than 50% and optimum design of LCA was accomplished.

1 まえがき

製品のプロトタイプ作製や量産立ち上げに、ゲートアレイに比べ短時間で開発可能な FPGA が多用されている。FPGA 開発は、回路図入力によるボトムアップ的設計作業が一般的であるが、ゲートアレイ開発における Hardware Description Language (HDL) によるトップダウン設計に比べ、最適設計（仕様に定められた機能を冗長度なしで、過不足なく実現すること）の保証に限界がある。そこで、FPGA 設計における最適設計の保証、短時間開発、および量産時の FPGA からゲートアレイへのリターゲットを目的に、HDL による設計手法の FPGA 開発への適用を図った。今回、多種の FPGA の中から、設計資産と開発機会の多い Xilinx 社の LCA を対象に、LCA とゲートアレイ間の回路接続情報変換手法を考案し、HDL による開発環境の統合を実現した。

2 HDL による開発環境の統合

Fig. 1 (A) (B) (C) は従来の LCA 開発フローである。設計者は LSI の仕様書から回路図作成までを主に紙と鉛筆で行なっている。この手法では、設計の上流工程である、回路図作成までの最適設計を保証するために、大きな労力と長い時間が必要で、しかも十分な保証には限界がある。そのため、(C) の論理検証時に誤りが発見されると、誤りが仕様書・機能分割・回路図生成のいずれに起因するのかという原因の特定と、修正が非常に大変である。

Fig. 1 (D) (E) (F) は HDL を用いたゲートアレイの開発フローである。今回使用した Verilog-HDL（米国 Cadence 社のデジタル ASIC 開発用 HDL）では、機能をレジスタ・トランスファ・レベル（以下 RT レベルと呼ぶ）で表現できるため、機能分割以降の設計工程を CAD

* 技術研究所 1G ASICGr.

** 技術研究所企画室

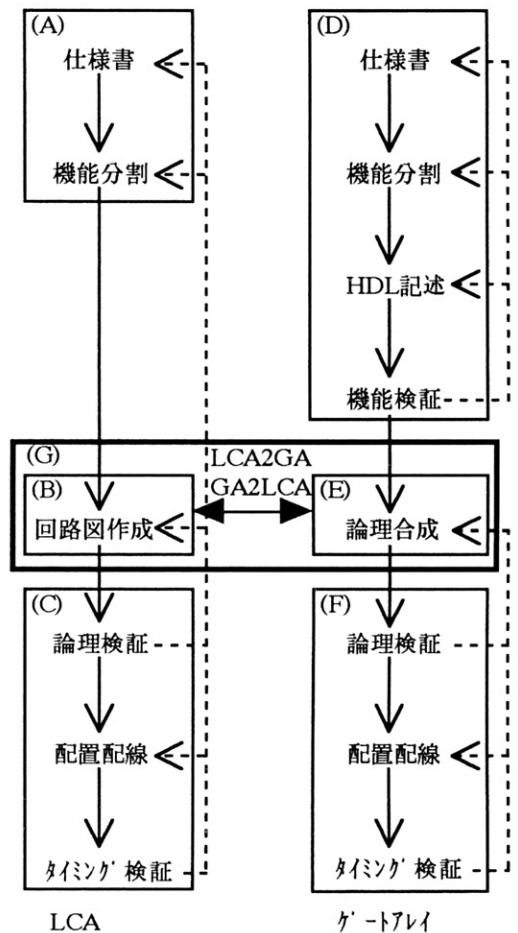


Fig. 1 Design flow of ASIC

上で行なうことができる。また、RT レベルの機能検証、論理合成後のゲートレベルの論理検証、配置配線後のタイミング検証と段階を踏んで行えるため、誤りがあっても早期に原因を特定できる。

このような2つの開発フローを (B) と (E) 間で結びあわせることで、

(1) Fig. 1ゲートアレイ設計フローの仕様書から機能検証までの上流工程 (D) を LCA 開発に適用することとなり、上流工程での最適設計と、開発期間短縮を図ることができる (D→E→B→C)。

また、

(2) 過去の LCA 設計資産の活用や、プロトタイプ作製から量産への移行に際しては、逆に (B) の LCA の回路図をもとに、(F) の論理検証から遅延検証までを行い、ゲートアレイにリターゲットすることが可能となる (B→E→F)。

そのためには、LCA の回路図作成ツールとゲートアレイの論理合成ツール間で回路接続情報の受け渡しをすれば良い。このデータ受け渡しは、双方のツールが LCA とゲートアレイ両者のシンボルライブラリを持っていることが前提である。しかし、現在は論理合成ツールが参照する、LCA のシンボルライブラリが供給されていないため、データの受け渡しができない。そこで、(G) にしめす回路図作成ツールと論理合成ツール間で、LCA からゲートアレイ (以下 LCA 2 GA と呼ぶ) と、ゲートアレイから LCA (以下 GA 2 LCA と呼ぶ) の双方向の回路接続情報変換ツールを作成し、相互のデータ授受を可能とした。

3 回路接続情報変換の課題

3.1 変換に用いるファイルフォーマット

回路図作成ツール、論理合成ツールは他のツールとインタフェースするための、各種入力データフォーマットを持っている。その中から、変換ツール実現に有効と考える、以下の3種について検討した。

(1) EDIF

論理合成ツールと回路図作成ツール双方で読み書きできる。しかし、EDIF は回路図生成のための情報を含まないため、回路図作成ツールが回路図を生成できない。このため、ゲートアレイから LCA への変換後、論理検証が不可能である。

(2) MENTOR-Do

論理合成ツールから回路図作成ツールに回路図情報を受け渡す (E→B) が可能なファイルフォーマットである。しかし、論理合成ツールが MENTOR-Do ファイルを読み込めないため、回路図作成ツールから論理合成ツールへのデータ受け渡し (B→E) は不可能である。

(3) MIF

回路図作成ツールから論理合成ツールにデータを受け渡す (B→E) が可能なファイルフォーマットであり、回路図作成ツールの標準インタフェースでもある。しかし、論理合成ツールが MIF ファイルを出力できないため、論理合成ツールから回路図作成ツールへのデータ受け渡し (E→B) は不可能であ

る。

以上から、GA 2 LCA には MENTOR-Do フォーマットを、LCA 2 GA には MIF フォーマットを採用することにした。ただし、論理合成ツールと回路図作成ツール間では、特殊記号 (たとえば、バスネットを表す「`]`」等) の扱いにわずかな違いがある。

3.2 GA 2 LCA のデータ処理

Fig. 1 の (E) から (B) へのデータの受け渡しは論理合成ツールが出力する接続情報から、回路図作成ツールが回路図を正しく生成できることがキーであり、以下にしめす3つの課題がある。

(1) 相互のシンボルサイズとピンの位置の相違

ゲートアレイと LCA のシンボルライブラリでは、回路図に用いるシンボル (論理を表す図形記号) のサイズとピンの位置が異なるため、単に等価な機能を持つ LCA のシンボルに置き換えただけでは、シンボルが重なり合ったり、配線が交差するために回路図チェックが困難となる。

(2) 負論理出力を有するシンボルの扱い

ゲートアレイのレジスタは、一般に正論理と負論理の2つの出力を持っているが、LCA では正論理出力のみである。このため、負論理出力を使用した回路では出力部の変換処理が必要である。

(3) 特殊記号の扱い

特殊記号はツールにより扱いが異なるため、変換先のツールにあわせて置き換える必要がある。

3.3 LCA 2 GA のデータ処理

Fig. 1 の (B) から (E) へのデータ受け渡しは、LCA 独自の回路構成をゲートアレイの接続情報に、正しく変換できることがキーであり、以下にしめす4つの課題がある。

(1) MIF ファイル生成時の制約

回路図作成ツールで MIF ファイルを生成する際、使用するシンボルの種類を指定する必要がある。これは LCA が階層的に特殊なシンボルを使用して回路図を表記しており、生成時にゲートアレイと整合をとるためである。

(2) LCA のグローバルリセットの扱い

LCA には、パワーオン時に全てのレジスタを初期化するための、グローバルリセットと呼ばれる機能がある。このため、LCA 回路に用いるレジスタには、特にリセット入力をもうけないのが一般的である。ゲートアレイにはこのような機能がないため、全てのレジスタをリセット付きに置き換え、グローバルリセット回路を追加する必要がある。

(3) 負論理入力を有するシンボルの扱い

LCA には、回路規模や遅延時間を改善するために、負論理入力を有するシンボルがある。ゲートアレイには、負論理入力を有するシンボルがないため、変

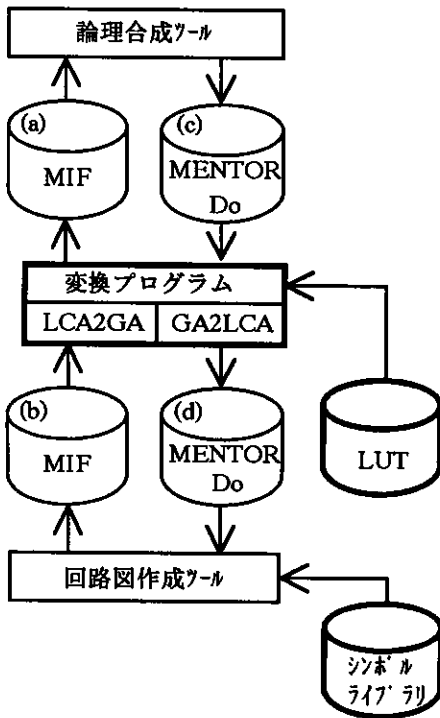


Fig.2 Data conversion flow

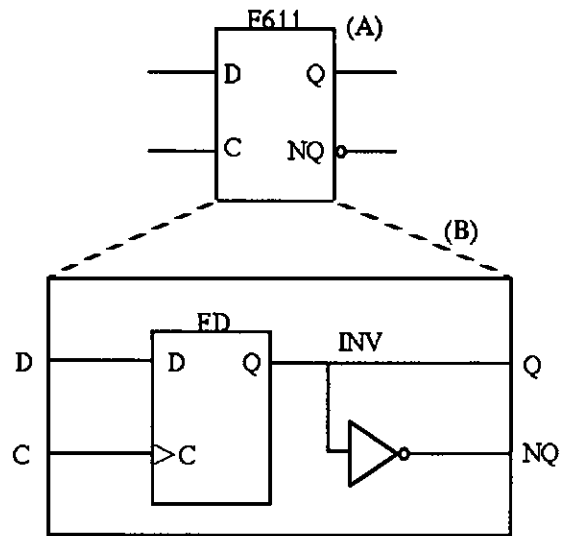


Fig.3 Example of the equivalent symbol

(A) : ゲートアレイ, (B) : LCA

LUT			
// buf //			
buf	b1n	0000000	
inv	v1n	0000000	
// and //			
and2b2	r2n	0000000	
and3b3	r3n	0000000	
// nor //			
// LCA GA	abcdefgh		
nor4b3	r4n	1110000	
nor5b3	r6b	1110010	
nor5b4	r6b	1111010	

a~f: LCAの入力が負論理であることをしめすフラグ '1'で負論理
 g :余っているゲートアレイの入力の数
 h :ゲートアレイの出力を反転することをしめすフラグ '1'で反転

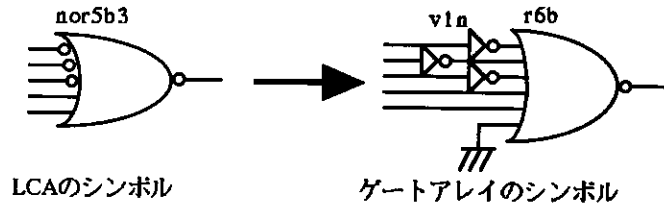


Fig.4 Format of LUT

換の際に入力にインバータを追加する必要がある。

(4) 特殊記号の扱い

3.2(3)と同様に、置き換えを行う必要がある。

4 回路接続情報変換ツールの開発

4.1 変換フロー

3章で述べた課題を解決し、Fig.1 (G) の LCA 2 GA、GA 2 LCA を実現するため、Fig.2 にしめす回路接続情報変換フローを考案した。

- (1) ゲートアレイから LCA への変換は、次のようにして行う。機能検証された HDL ファイルを、論理合成ツールでゲートアレイの回路に変換した後、MENTOR-Do ファイル (c) を生成する。次に変換

プログラムで特殊記号を置換し、出力した MENTOR-Do ファイル (d) を回路図作成ツールに入力する。回路図作成ツールはシンボルライブラリを参照して、LCA の回路を生成する。

- (2) LCA からゲートアレイへの変換は、回路図作成ツールでゲートアレイのシンボルに、置き換え可能な LCA のシンボルを指定し、MIF ファイル (b) を生成する。次に、LCA とゲートアレイ間でのシンボルの対応関係を記載した、ルックアップテーブル (以下 LUT と呼ぶ) を参照し、変換プログラムでゲートアレイの MIF ファイル (a) に変換する。このファイルを論理合成ツールに入力し、ゲートアレイ回路を生成する。

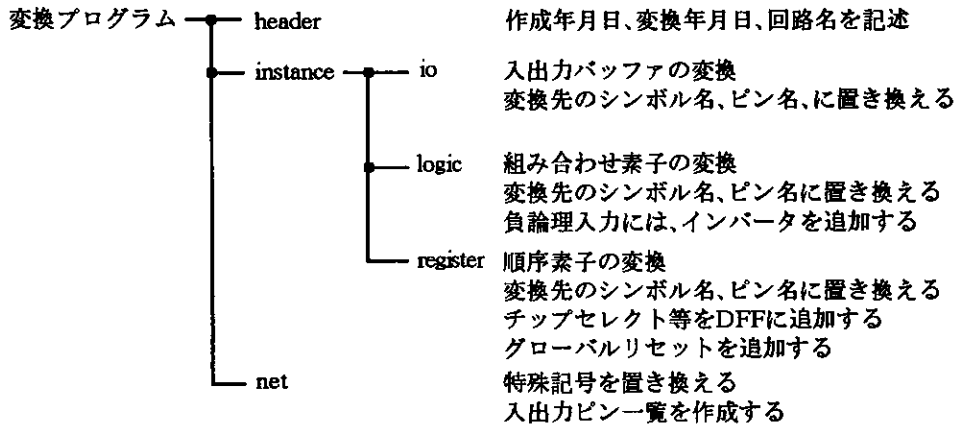


Fig. 5 Structure of conversion program

4.2 変換ツール

前節で述べた回路接続情報変換フローを実現するため、シンボルライブラリとLUT及び変換プログラムを作成した。

- (1) シンボルライブラリは、回路図作成ツールが参照するもので、個々のシンボルはFig. 3のような構成となっている。(A)はゲートアレイのシンボルを置換するための等価シンボルで、サイズ、ピンの位置が同一である。下位階層(B)に実際のFDやINV等のLCAのシンボルから成る回路を持っており、LCAとしての論理検証が可能である。
- (2) LUTは、Fig. 4に示すように、LCAのシンボルをゲートアレイのシンボルに置き換える時の、シンボル間の対応関係を記述したテーブルであり、置換シンボル名と入出力部処理フラグから構成される。この例では、ゲートアレイに5入力のNORシンボルがないため、nor 5 b 3 (5入力)をr 6 b (6入力)に置換し、a~hの記述から、r 6 bの3つの入力にインバータv 1 nを追加し、余った1つをグラウンドに接続する処理を行っている。hは、出力をさらに反転する必要がある場合に使用する。
- (3) 変換プログラムは、
 - ① LUTを参照して、MIFファイルフォーマットの回路接続情報を、LCA用からゲートアレイ用に変換
 - ② 双方向の変換処理において特殊記号を置換の2つの機能を持っている。

Fig. 5に、その構成と各部の役割をしめす。この変換プログラムはC言語で開発した。

5 むすび

LCAの最適設計保証、開発期間短縮と量産時のLCAからゲートアレイへのリターゲットを目的に、HDL設計手法を適用した開発環境を構築した。回路図作成ツールと論理合成ツール間でのデータ授受を可能とするツールを作成することにより、開発環境の統合を実現した。特に、回路表現が相互に異なるため等価シンボル、LUTを考案し対処した。変換処理は、C言語によるプログラムを作成し、処理の自動化を図った。

統合した本環境をLCA開発に適用した結果、ゲートアレイと同様のトップダウン設計による最適保証と、従来の1/2の設計期間短縮を実現でき、その有用性を確認した。したがって、今後のLCA設計は、ゲートアレイ設計と同一のHDLを用いて論理合成を行い、回路図作成以降のステップへ進む本手法を採用することとした。

今後の課題として、以下の2点がある。

- (1) LCA固有のマクロ変換を可能にする。
- (2) 他FPGAとゲートアレイとの変換を実現する。