計算機アーキテクチャ 113−31
（１９９５．８．２５）

耐故障性マルチプロセッサのための SOFT チップの設計と評価

高西 裕治, 玉木 淳一, 上原 稔, 森 秀樹
東洋大学工学部情報工学科

本論文ではストリーム通信に基づく並行オブジェクトモデルのためのプロセッサ接続方式 (SOFT) を FPGA(Field Programmable Gate Array) で実現するための試作チップの設計について述べる。SOFT は実行したい計算を 3 つのプロセッサで同時にパイプライン処理を行ない、その演算結果の多数決をとることでフォールトトレラントを実現するものである。本研究ではそのための回路設計を行ない、回路シミュレータで動作の確認をして耐故障性に対する評価と FPGA のためのゲート数を検討する。

Design and Evaluation of SOFT chips for Fault Tolerant Multiprocessors

Yuji Takanishi, Junichi Tamaki, Hideki Mori, Minoru Uehara
Department of Information and Computer Sciences,
Toyo University,kawagoe 350,Japan
E-mail:takani@mo.cs.toyo.ac.jp

In this paper ,we describes the design of SOFT(Stream Oretnted Fault Tolerant) to implement concurrent objects model based on stream connection. SOFT emlpoys fault tolerant pipeline processing which realizes fault tolerant features by voting of each computed results in three cells. In addition, we have confirmed the operations and have evaluated both fault tolerant perfomance and the number of gates for FPGA implementation.

1 まえがき

現在 VLSI 技術の発展により、多数のプロセッサで構成される並列処理システムの研究が盛んに行なわれている。そのようなシステムではそれぞれのプロセッサと接続するパス数が増大し、それによってデータの故障に影響を及ぼす危険性がある。また、プロセッサが 1 つぐらい故障しても並列処理が正しく実行されないと他のプロセッサが無駄になってしまう。これらのことから故障回避を実現することは、並列処理システムを実現するためにもプロセッサとパスに対してのフォールトトレランスが重要になる。そこで本研究では、多数決と信頼度評価を用いたストリームに特徴をなすフォールトトレラントなプロセッサ接続方式 (SOFT)[1][5] を用いたチップの設計を行ない回路シミュレータで動作確認、面積、ゲート数、消費電力を検討する。現在、本研究では FPGA (Field Programmable Gate Array) を用いて設計した SOFT のチップ機能を確認して最終的に ASIC(Application Specific Integrated Circuit) 回路として実現することを目標としている。FPGA はゲートの組合せを何回でも書き換えることができるので試作機からの進化が容易である。そこで本論文では主として FPGA による実現について述べる。

構成は 2 章では SOFT について述べ、3 章は設計における問題と解決策、4 章で設計についての検討、5 章でまとめを述べる。

2 SOFT

2.1 SOFT について

SOFT(Stream Oriented Fault Tolerant) アーキテクチャは多数決と信頼度情報を用いたデータ比較を (信頼度型比較) を行なう。その情報は直前に行なわれた計算結果との一致をとることにより生成される SOFT では、同じ計算を 3 つのセルで同時に並列処理し、その演算結果の多数決をとることでフォールトトレラントが実現される。ここでセルとは、CPU の機能とマスク部を持ったプロセッサである。その処理ステップはパイプライン処理に基づき、その直前ステップにおけるグループ内の 3 つのセルから 2 つのセルが次回の処理ステップで重複して使用される。よって SOFT アーキテクチャは三重化を用いているにも関わらず、総セル数は、基本セル数と同程度に押えられている。そのアーキテクチャは、それぞれの、セルから 8 方向にパスを張り出している。そのパスは、データの行き帰り別のパスでシリアル転送するため総計するとパスは 16 本分必要となる。なぜシリアルかは 3 章で述べる。マスク部は Voter、比較回路、信頼度評価の部分を 1 つずつ含んでいる。(マスク部の概要は 2.2 で述べる) それぞれのセルは、続いて起こるグループ内のセルである近隣の 3 つのセルに対してデータを送る。そしてそれぞれのセルは、データを受けとり個々の計算を実行する。Voter 部では、先に行なわれたグループ内のセルから送られてきた 3 つの入力データから多数決を行なうことで、1 つのデータを選択する。信頼度評価部では、最も新しい個々の CPU 計算結果が信頼できるかを信頼度情報として生成する。信頼度比較はセルに 2 つのデータしか入力されず、かつ比較部において一致がとれなかった場合に対して信頼できる情報の選択に使用する。SOFT では、このような格子型プロセッサアレイに (プロセス) を割り付けてフォールトトレラント計算を実行する。

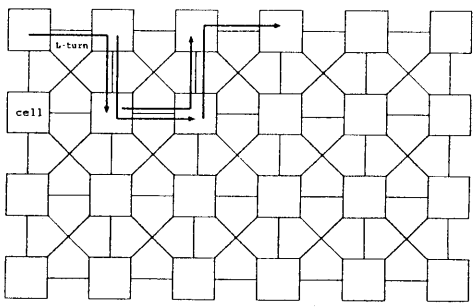

図 1: ストリーム計算のためのフォールトトレラントプロセッサ方式

次に 3 つのセルの基本的組合せとなる L-turn[5] について図 1に示す。L-turn に含まれる 3 つのセルは、それぞれが並列に処理を行なう。そして、L-turn 内の 3 つのセルで処理が行なわれると、3 つのセルの内 2 つが次の処理でも再び使われることになる。このアプローチは、先に処理された L-turn からの

出力データを次の L-turn の各セルが多数決することを基本としている。つまり、2 つ以上の入力データが一致すればデータは適切に処理され、計算は続けられる。さらに、それぞれの処理を行なうセルは、自分自身で行なった計算結果が正しいと判断し、入力されたデータの多数決結果と比較を行なうことにより、その比較結果を信頼度としてデータ共に出力する。

L-turn には type1、type2 の 2 種類がある。type1 はすべてのセルで多数決が行なわれる。type2 では多数決を行なうか、信頼度を比較しデータを選ぶかは、セルの位置によって決まる。まず、type1 は CPU の計算処理は、L-turn ないの 3 つのセル全てで行なわれる。図 2の場合を例に挙げると、セル 2、3、4 の時である。L-turn 内の 3 つのセルは、3 入力多数決を行なう。それぞれのセルが受けとる入力データと自分自身のデータである。

図 2のセル 2 の場合、先に行なわれた type2 内のセル 1、3 からデータを受けとり、セル 3 と 4 の場合は、セル 1、2 からデータを受けとる。つまり、type1 内のそれぞれのセル (2,3,4) は、先に行なわれた type2 内のセル (1,2,3) からデータを受けとる。次の CPU による計算のためにセル 2、3 は、自分自身の前の計算結果と多数決の結果の比較を取ることにより、信頼度を生成する。type2 は CPU の計算処理を L-turn 内の 3 つのセルで行う。つまり、図 2の type2 の場合を例に挙げるとセル 3、4、5 のときである。type2 において、L-turn の後側のセル 3、4 は先に行なわれた type1 中の 2 つのセルから入力と、自分自身が直前に行なった結果の 3 つにより、3 入力多数決を行なう。つまり、type2 内のセル (2、3、4) からそれぞれデータを受けとる。type2 の先頭に配置されているセル 5 は、先に行なわれた type1 のセル 3、4 からデータを受けとり、その 2 入力データが比較部で一致がとれなかった場合に、信頼できるデータを 1 つ選択するためにデータ信頼度を使用する。次の CPU による計算処理のために、2 つの後側のセル 3、4 は、自分自身の前の計算結果と多数決結果の比較を取ることにより、信頼度を生成する。以上が L-turn である。

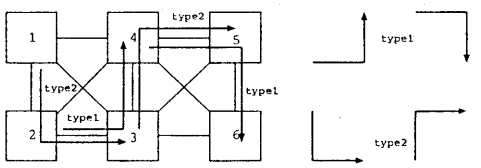

図 2: L-turn の type 図

2.2 マスク部の概要

マスクは故障の影響が外部に現れないだけで故障自体は除去されない。よって長い時間が経過すると故障が蓄積される。そこで用意された冗長がつきてシステムの故障につながるといった欠点を持つ。しかし外部に対する正常な動作が途切れないので、リアルタイム性が求められる場合に適する。またシステムの一部のある活動要求なしで耐故障性を達成できるという利点を持つ。本研究のアーキテクチャはデータが流れてくるというところからリアルタイム性が求められるのでマスクを用いている。

次にマスク部の全体の動きを述べながら各部分の回路について説明する。概要図は図 3に示す。マスク部は TMR(Triple Modular Redundancy) を基にデータ信頼度を取り入れたものである。その動作は入力部 (Data Input) で、他のセルの出力と自己の出力を入力されて来るデータとして 3 つ選ぶ。この制御は CPU から来る信号で決める。また入力部ではシリアルデータをパラレルデータに変換すると共にパリティチェクを行なう。こうしてそろえられたデータとは多数決回路 (Voter) とデータ選択回路 (Data Selector) に送られる。データ信頼度も同じようにしてデータ信頼度判定回路 (Data Reliability Check) へ送られる。つぎに、多数決回路 (Voter) が、ある 2 つのセルからの入力と前回の自セルの演算結果で多数決を行なう。その後、前回の自セルの結果と Voter の出力を比較回路 (Compare) で比較して自己のデータ信頼度を判定する。その時の信頼度は自セルのデータ信頼度として Register に確保される。この自己データ信頼度は CPU の計算結果データと共に出力部を経て他セルに出力される。多数決非成立時のときはデータセレクタ (Data Selector) とデー

タ信頼度判定回路 (Data Reliability Check) で、高信頼度のデータを選択して CPU に送られる。先に説明したように Type2 の L-turn の先頭のセルは入力データが 2 つなので多数決回路が使えない。そこで比較判定回路 (Match) を用いて 2 データを比較して一致したらそのままデータを CPU に送りそうでなかったら 2 データのそれぞれの信頼度を比べて高い方のデータを CPU に送る。出力部はでは計算結果を受けとりパラレルデータをシリアルデータに変換して近傍のセルへ出力する。以上がマスク部の動作である。

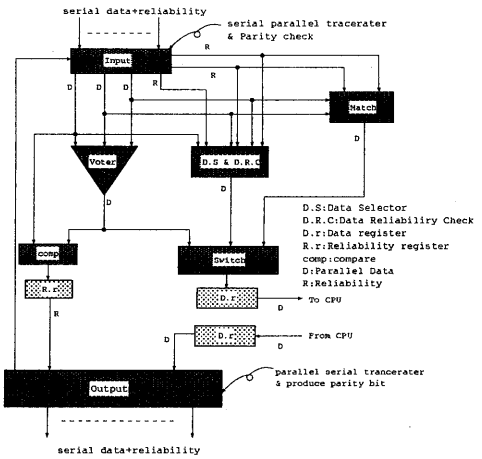

図 3: mask 図

3 設計における問題点と解決策

ここでは各部分での設計で起きた問題点を挙げて解決策を述べる。

3.1 セル同士の通信での同期

SOFT は 3 つのセルで同時に実行したい計算をする。そのため計算する時に他のセルからデータを受けとって計算できる状態に 3 つのセルがなっていなければならないといった同期の問題がある。さらに論ずると同期は 2 段階で行う必要がある。1 つは SOFT と CPU 間のタイミングで、これはコンパイラの解析に一任することにした。また、セル同士の通信でデータを受ける側がデータを送られてくるまで待っているとデータが故障で送られてこなかったときにそこで処理が止まってしまう。そこで SOFT では 1 つの処理を終えてから 1 データ分送られて来る時間だけ待って次のマスク作業を行なう様にした。こうすることで故障により処理が停止することがなくなる。もう一つは SOFT 間の各データ転送の同期で調歩同期通信を用いて同期を取ることにした。調歩同期通信は伝送する 1 キャラクタ信号の前にスタートビットを付加することで同期をとる方式である。受信側では 0 の次からデータとして受けとる。データが送られてこないときは 1 が送られて来るのでデータとして受けとらない。本チップでは 0 をスタートビットとして転送し、そのあとの転送をデータとした。

またセル同士の通信をパラレルで行ないところだが 1 つのセルに対して入力、出力の線 2 本でこれが 8 方向近傍のセルとつながっているのでこれをパラレルにするとデータ 16 ビットと信頼度 2 ビットとパリティビット 2 ビットで合計 20 本。セル間の配線は計 160 本となり、かなり多い配線になる。もしこれを実際に作るとデータ転送中の故障の原因になったり手で配線するので失敗する危険性も高い。そこで今回のチップ設計は動作確認が最優先なのでシリアル転送を用いることにした。

3.2 CPU とのつながり

SOFT には CPU とマスク部をつなぐ部分が必要である。今回の CPU は SOFT と一緒に 1 つの FPGA に納まらないため市販のものを用いることにした。また CPU は最終的にセルを数十台以上用意することを考えてコストパフォーマンスに優れた SH7034[2] を用いる。今回の設計では実装が用意になるように SH に依存した設計を行なうが SOFT 自体は本来どのような CPU と接続することも可能である。

以下に今回の CPU とマスク部の関係を述べる。SH は 16 ビットの入出力ポートを 2 つ持っている。そこで、SOFT の CPU とのインターフェイスも 16 ビットにした。また 1 つのポートを SOFT の制御と SOFT 状態信号としてもう 1 つのポートをデー

タ入出力用とした。CPU からの制御はセルの入力が 8 方向のどの入力方向を使うか示すのと自己の値を使うかを示す信号としてと入力データが 3 方向か 2 方向かを示す信号とデータ用のポートが入力か出力かを示す信号とパラレル・シリアル変換をして外部に出力を始めることを促す信号の計 12 ビットである。また SOFT の状態信号は多数決終了信号とパラレル・シリアル変換終了信号の 2 ビットである。残りの 2 ビットは設計をしていく上で出てくる問題のために残しておくことにする。これによりセルの内部構成が決定でき CPU とマスク部の関係が決定した。ポートとマスク部の関係を図 4に、制御用のポート (port A) を図 5に示す。

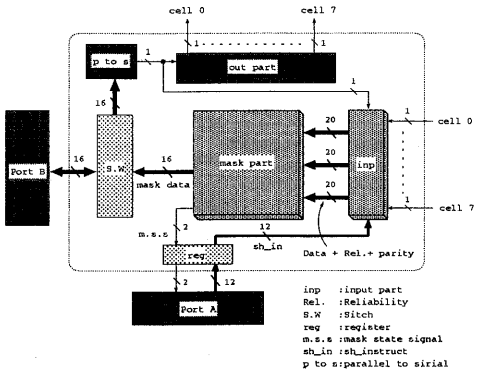

図 4: CPU とマスク部の関係図

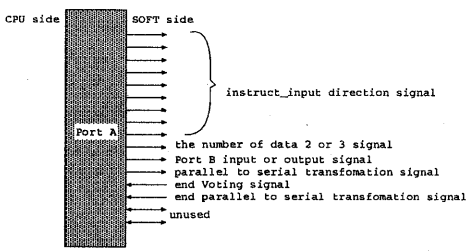

図 5: Port A の仕様

3.3 パス上での故障

ここでは SOFT にパス上での故障に備えてパリティチェック機能を追加する設計について述べる。まず入力部でデータ 16 ビット信頼度 2 ビットパリティビット 2 ビットとして制御で決まった方向のセルからデータを受けとる。パリティは奇パリティとした。それはパリティビットを 2 ビットにすることでスタック 0、1 故障も発見できるようにするためである。図 6に示すようにデータ、信頼度、パリティビットが 1 になるスタック 1 故障だと 1 が偶数個あるのでエラーとなる。またデータ、信頼度、パリティビットがすべて 0 になるスタック 0 故障だと 1 がないので奇パリティでないのでエラーとなる。このようにエラーが発見できるが、もしパリティビットが 1 ビットだとするとスタック 1 故障では 1 が奇数こなのでエラーを発見できない。以上のことからパリティビットを 2 ビットとした。

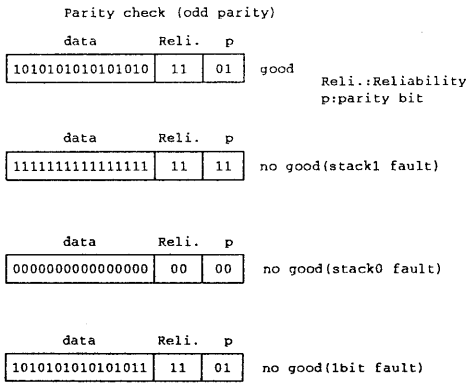

図 6: Parity check 判定

図 6に示すような故障に対して発見できる。ここで故障が発見できた時に 11 なら 10、10 なら 01、01 なら 00 と信頼度を 1 ビット分下げることにした。また、自己のセルの信頼度はマスク結果と自己のデータが一致するたびに 00 なら 01、01 なら 10、10 なら 11 といったように信頼度を 1 ビット上げるようにした。

3.4 Voter 回路の問題

ここで Voter 回路部について示し入力データが 3 つ異なったときの Voter の問題点について述べて解決策を示す。今データを 1 ビットとして 3 入力をそれぞれ A<0>B<0>C<0>とすると出力 Q は

$$Q=A<0>B<0>+B<0>C<0>+A<0>C<0>$$

となる。次にデータサイズ 3 ビットで入力が 3 つ異なった場合を考える。A<2:0>が 100、B<2:0>が 001、C<2:0>が 111 だとすると先に述べた式より出力は 101 と 3 つの入力データと異なる値になってしまう。これでは多数決どころか違うデータを正しいと認識するといった問題がある。そこで図 7が示すような回路をつくってマスク部の最終的にデータを決める Switch 回路に入力が 3 つ異なるかどうかを示す信号を送り 3 つ異なる時は Data Selector 出力を選ぶようにした。図 7に示した回路は 3 ビットデータなので後は同じ方法で用いるデータサイズの回路を組めば良い。以上のようにして入力データが 3 つ異なってもマスク部が機能する様にした。

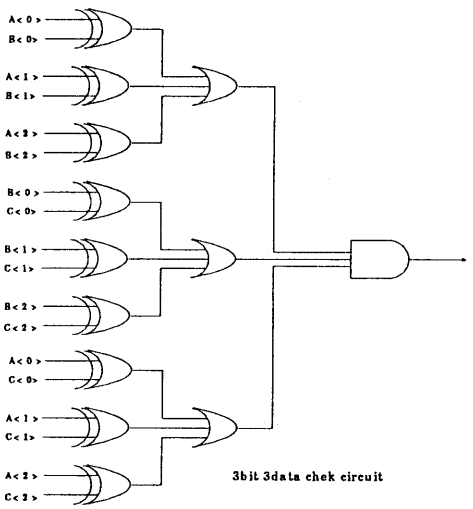

図 7: 3bit 3data check circuit

3.5 設計の方法

本研究では PARTHENON[4] を用いて設計し動作の確認を行なった。PARTHENON を選んだ理由を示すために以下に PARTHENON の特徴を述べる。PARTHENON は NTT で開発されたハードウェア設計支援システムである。PARTHENON では動作記述言語 SFL を用いて作成する回路を書くとシミュレーションによる動作の確認が行なえる。また Verilog-HDL、VHDL などの他のハードウェア記述言語と比較すると、それらがイベント駆動に基づいているのに対して SFL は動作の流れ (要求駆動) を基としているので記述しやすい。さらに、Verilog-HDL、VHDL がシミュレーションのために開発されたのに対して、SFL は単一のクロックのハードウェアに限定しているので記述力で SFL が良いと考える。さらに、SFL ソースから LSI の配置配線プログラムや FPGA マッピング・ツールへの入力となるネットリストを作成でき、これを XLINX の FPGA 用に変換する nld_xnf コマンドを実行するだけで FPGA にマッピング可能なフォーマット XNF になる。以上より今回の設計は PARTHENON を用いることにした。また設計で使用する FPGA は XILNX4010 である。

3.6 デバッグのための端子

マスク部を FPGA にマッピングするのに成功した時に動作の確認をするためにデータの出力端子以外にも端子を参照できるようにしておく必要がある。そこで今回の設計では表 1に述べる端子をデバッグのために用意する。こうすることにより動作が正しくいかなかったときにマスク部のどの部分が正しく動いてないか即座に発見でき修正のスピードを速くすることができる。今回使用できる FPGA は 200 本まで端子を使用できる表 1に示した端子を合計すると 140 本になる。よって端子の数では FPGA にマッピングできることは明らかである。また残りの 60 本については今後 FPGA で動作を確認する時に出てくる問題について参照できるように残しておく。問題が起きた時に必要な端子を SFL で記述して FPGA にマッピングするだけで見たい端子が使えるようになる。

端子名	bit 数	機能
outd	16	vote 結果
outc	2	信頼度生成値
outds	16	selector 結果
pd0〜2	18	シリアル・パラレル変換結果
md	16	マッチ結果
sto	2	SOFT 状態信号
outk	1	3 入力不一致
pk	1	parity check ok

表 1: デバッグ端子表

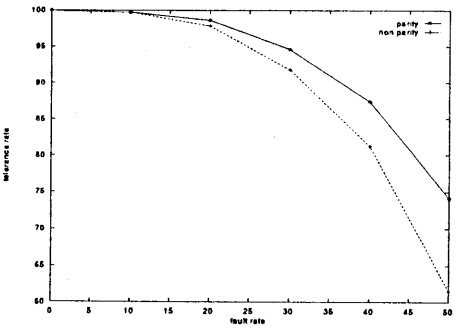

図 8: シミュレーション結果のグラフ

4 評価

ここでは今回新たに加えたパリティチッェク機能の有無についてパス上でデータ故障が起きた時の回避の度合を比較検討する。また PARTHENON で設計した回路のゲート数を求めて FPGA にマッピングできるか否かを調べる。まずパリティチェックについて考えると 3 つの同じ 16 ビットデータを用意し、それぞれにランダムで 2 ビットの信頼度をつける。その後この 18 ビットの中に 1 がいくつあるか数えてパリティビットを 2 ビット加える。次に 3 つのデータに対して違う初期値を与えて 0%〜50%までの故障率でデータを故障させる。その 3 つのデータをマスク部に入力し、マスク部の出力を見て故障を回避できているかそうでないかを確かめる。まず、SFL で以上のような故障を発生するライブラリを作り、これを SOFT に組み込んで評価を行なった。各故障パーセントに対して 1000 回シミュレーションして 1000 回正しければ 100% としてマスク部が何回正しい答を出したかを比較の対象とする。例えば 1000 回シミュレーションした内 997 回正しい値をマスク部が出力すれば 99.7%の正常動作となる。

グラフ 8を見ると故障が起きる確率が 10%ぐらいのときはパリティチェック付きと無しの差がないが確率が 40%当たりから差がでてきていることがわかる。少しでも故障を回避できるほうが良いと考えるのでパス上のデータ故障に対してはパリティチェックを付けたほうが良いと考える。パリティチェック付きと無しのデータ故障に対する差を述べたが、パリティチェック付きと無しでハードウェアではどれぐらい差があるのかを次に述べる。SFL で記述した回路を PARTHENON で面積 (area)、消費電力 (power)、ゲート数 (gates) を求めたところ表 2のようになる。ゲート数で比較するとパリティチェック付きが 4665

項目	parity	non parity
power	2973.79	1995.35
area	378.059	300.46
gates	4665	3778

表 2: 回路の合成結果

ゲート、無しが 3778 ゲートで差は 887 ゲートである。そこで故障に対しての差を先のグラフで見て述べたようにパリティチェック付きの方が故障に対して強いことがわかる。このことから 887 ゲートぐらいの差ならパリティを使った方が良いと考えられる。以上のことから今回の設計で加えたパリティチェックはパス上でのデータ故障に対してマスク部の性能を向上させた。今回使用できる FPGA(XILINX4010) はゲート数 1 万、端子数は 200 本使用できる。本研究で作成した回路は先に述べたようにパリティチェック付きで 4665 ゲートで端子数は 140 本なのでこの 2 点については FPGA にマッピングできることは明らかである。また PARTHENON で作った回路の標準ネットリストフォーマット NLD/PCD を XILINX[3]

のネットリストフォーマットXNFに変換する。ここで問題が起きた。それは双方向端子でXLINXではこれに相当するライブラリを用意していない。そこで今回は変換を成功させてゲート数を確認したかったので双方向端子を入出力端子に分けて変換した。よってXNFにすることでPARTHENONでは仮想のゲートであった回路をFPGAにマッピングできるフォーマットにしたことになる。

5 まとめ

本論文では、SOFTチップの設計をして、SFLで記述した回路を耐故障性の評価を行なった。またゲート数や端子数でもFPGAにマッピングできるかを確かめ実際にこれを行なった。今後の目標としてはFPGAへのマッピングを行なってASICに落とすことである。ASICにしてからバグが見つかったのではまた作り直すのにとてもコストがかかる。そこでFPGAの段階でFPGAとCPUの評価ボードを接続させ、その接続したボードでストリームを使ったパイプライン計算とフォールトトレラントが実現できているかをテストベクトルを使って確認する。この段階でバグを発見したら設計を修正してFPGAにマッピングし直し正しく機能するまでこれを繰り返す必要がある。また、本論文では動かすのが目的なので入手できたCPUの評価ボードを使ったが、いずれはいろいろなCPUに対してSOFTが対応できるように設計する必要がある。

謝辞

PARTHENONを使用させていただいたNTTに感謝致します。

参考文献

[1] Hideki Mori,Minoru Uehara「Stream Oriented Fault Tolerant Architekucture」
In International Conference on WAFER SCALE INTEGRATION,IEEE,Jan 1994

[2] 「SH7034 ハードウェアマニュアル」
発行 株式会社 日立製作所 半導体事業部

[3] 「The Programmable Logic Data Book」The Programmable Logic Company,1994

[4] NTTコミュニケーション科学研究所 NTTデータ通信株式会社 著
「第二回 PARTHENON 講習会 テキスト」
パルテノン研究会

[5] 田口 英伸 「SOFTにおけるアクタ割り付けアルゴリズム」情報処理学会第50回全国大会,March,1995