マイクロコンピュータとワークステーション 62－4
（1990. 6. 22）

トロン仕様準拠32ビットMPU M32／100をコアとするASSP開発

中尾 裕一+ 大木 正司+ 北上 尚一+ 是松 次郎++
+ 三菱電機株式会社北伊丹製作所
++三菱電機株式会社LSI研究所

トロン仕様に基づく32ビットマイクロプロセッサをコアとして、複数の周辺機能を内蔵したASSP（特定用途向け標準製品）を開発した。製品展開を迅速におこなうために以下の設計手法を採用した。スタンダードセル手法と汎用セルルータを用いて階層設計をおこなうとともに、テスト用切り換え回路の使用とモジュール毎の論理検証の実行とにより、モジュールの入れ替えに柔軟に対応できるようにした。また、簡単な入出力信号記述を用いた論理シミュレーション入力の生成と、論理のモデル化をおこなって、検証とテストパターン生成の工数を削減した。MPUコアのモデル化記述にはハードウェア記述を利用した。

A DEVELOPMENT OF ASSP WITH M32／100 MPU CORE BASED ON TRON SPECIFICATIONS

Yuichi Nakao+ Masashi Ohki+ Naoichi Kitakami+ Jirou Korematsu++
++ Mitsubishi Electric Corporation Kitaitami Works
++ Mitsubishi Electric Corporation LSI Laboratories

An ASSP has been developed , which includes a 32bit microprocessor based on TRON specifications as a core , and peripheral function units. We adopted a design method fit for rapid development of the ASSP series , as follows. A hierarchical design method using a standard cell and a general cell router , a test circuit , and a module-divided logic verification facilitate a replace of the functions. A simple signal descreption for logic simulation , and a model circuitry representing function modules have shortened the logic simulation time. In order to make the model of the MPU core , we adopted a hardware descreption langage .

1．はじめに

M32／ASSPは、TRONCHIP仕様[1]にもとづく32ビットマイクロプロセッサ（M32／100[2]）をコアにして複数の周辺機能を内蔵した、機能集積型のマイクロプロセッサである。ASSP（Application Specific Standard Products）という名称の示す通り、このLSIは応用分野をある程度限定したうえで、用途に応じて必要とされる機能をプロセッサに取り込んだものである。応用製品の性能（機能、速度、信頼性）を高め、またコストを低減させることを目指している。今回開発したASSPでは、高機能ワードプロセッサ市場を念頭においた。

ASSPの開発においては、どのような市場に限定し、どのような機能を内蔵するかということが最も重要な課題であり、製品展開の豊富さと開発の速さとが求められる。そのため設計手法としても効率的な階層設計が望まれる。階層化設計への第1の要求は、機能モジュールの資産化である。具体的には、論理またはレイアウトのレベルでモジュール化を進めチップをつくるための資産として再利用可能にすること、さらにテストベクタも機能毎にモジュール化しチップを検証するための資産として再利用可能とすることである。階層化設計への第2の要求は、LSI全体の動作、つまり機能モジュール毎の動作と機能モジュール相互間の動作とを容易にテストできるようにすることである。今回のMPUコアのようにモジュールの機能が高く動作が複雑である場合、チップ上の全論理を常に用いてテスタベクタ生成のための論理シミュレーションをおこなうことは、効率的ではないしその必要もない。第2の要求に対する解答の1つは、個々の機能モジュールのテストベクタ生成を目的とする論理シミュレーションの際には、他の機能モジュールの論理を単純なモデルで表現し、またモジュール相互間の論理検証とテストベクタ生成のための論理シミュレーションの際には、その検証に不要な機能モジュール内の論理はモデル化してしまうことである。

今回の報告では、M32／ASSPの実際の設計に際し、前記の2つの要求に応えるために用いたモジュール化の考え方と、論理検証およびテストベクタ生成の手法について説明する。

2．M32／ASSPの概要

M32／ASSPは、高機能ワードプロセッサ市場をターゲットにして、必要な機能を1チップ化したものである。M32／100コアを核にして、DMAコントローラ、割込みコントローラ、タイマ、シリアルインタフェース、ウェイトコントローラ、リフレッシュコントローラ、バス権調停回路を内蔵している。MPUの命令セット、レジスタセット等のアーキテクチャ、命令実行時間、割込み処理時間等の性能は、M32／100と全く同じである。

M32／ASSPの概要を表1に示す。

表1　M32／ASSP概要

MPUコア	32ﾋﾞｯﾄMPU（M32/100） 7MIPS/20MHz
DMAC	4ﾁｬﾈﾙ ﾃﾞｰﾀｱｾﾝﾌﾞﾙ,ﾌｧﾈﾘﾝｸﾞ機能付
IRC	14本のﾛｰｶﾙ割込み 3本の拡張割込み
タイマ	16ﾋﾞｯﾄx3本 8ﾋﾞｯﾄﾌﾟﾘｽｹｰﾗ
ｼﾘｱﾙｲﾝﾀﾌｪｰｽ	全2重ﾀﾞﾌﾞﾙﾊﾞｯﾌｧ 転送ﾚｰﾄ可変
ｳｪｲﾄｺﾝﾄﾛｰﾗ	0～5ｳｪｲﾄx4領域
ﾘﾌﾚｯｼｭｺﾝﾄﾛｰﾗ	12ﾋﾞｯﾄのﾘﾌﾚｯｼｭｱﾄﾞﾚｽを出力
ﾊﾞｽｱｰﾋﾞﾀ	MPU､DMAC等のﾊﾞｽ使用権調停
プロセス	1μmCMOS、2層Al
サイズ	15.25mm×13.7mm
パッケージ	179pinPGA

3．M32／ASSPの設計法

1）設計手法

今回の開発では、モジュールの設計にスタンダードセル手法を用い、得られたレイアウトをマクロセルとして汎用セルルータにより自動配置配線してチップを構成する方法を取った。ただしＭＰＵコアの部分に関しては、すでにでき上がっているレイアウトをそのままマクロセルとして使用した。スタンダードセル手法を用いた理由は次の通りである。

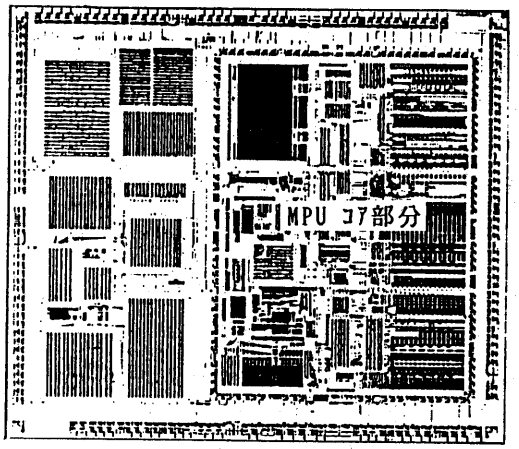
ＡＳＳＰ製品としての展開を考えた場合、新規機能の開発工期の短縮、既存機能モジュールの資産化、およびチップ面積の圧縮が求められる。面積を小さく抑えるためには、ハンドクラフト設計が優れているが、工期短縮、モジュール資産化という点からみると、スタンダードセル手法の方が優れている。マクロセルを組み合わせてチップを作る場合、マクロセル毎の縦／横比率、配置および配線の仕方が面積を決める重要なファクタであり、その点スタンダードセル手法だとマクロセル作成に複数回の試行が可能であることから、面積増大のデメリットは大きくないと判断しスタンダードセル手法を採用した。設計の資産としては、ＥＷＳ上に入力された論理図面と配置配線の優先順位情報とを残すことにした。

2）モジュール分割

Ｍ３２／ＡＳＳＰで用いたモジュール分割の考え方を説明する。

今回新たに設計した周辺機能は、もともと複数の機能に分かれており、機能をモジュールに分割して並行に設計することが容易であった。論理設計は７つの機能モジュールとパッド周辺の制御回路との８つに分割して並行に作業を進めた。各機能モジュールはそれぞれ共通のバスインタフェースでバスに接続されるとともに論理的に独立している。一方、レイアウトに関しては、マクロセルサイズにアンバランスがあるとチップを構成する上位階層の配置配線で無駄な部分ができてしまうことが予想されるため、個々の機能モジュールの論理量を勘案し、統合をおこなって６つのレイアウトブロックとパッド周辺の制御回路に分けて配置配線をおこなった。

Ｍ３２／ＡＳＳＰのチップ写真を第１図に示す。



第 1 図

テストの容易化に関しては、新規設計の周辺機能モジュールと既存のＭＰＵコアに対して異なる考え方をした。

周辺機能モジュールをテストするために、回路を切り換えてチップ外部から機能モジュールのレジスタへのバスアクセスを実現するモードを設けた。既に述べたように、各機能モジュールはバスインタフェースでバスに接続されるとともに論理的にほとんど独立している。テスト回路を追加した結果、機能モジュールを個別にテストすることができるようになった。一方、機能モジュール間の専用信号線の動作については、チップ外部からの直接的な制御や観測をおこなうための回路を特に追加しなかった。専用信号線につながる片方のモジュールのモードを固定し、一定の規則で加工された信号線を観測するにとどめた。その理由は周辺機能モジュールの論理が比較的浅く制御と観測が容易なので、面積と不具合発生率を増大させてまでテスト回路を追加する必要がないと判断したためである。

ＭＰＵコアに対する考え方はこれとは異なり、コアにつながる信号を直接チップの外から制御観測するためのテスト用回路を設けた。通常モード

ではコアと周辺機能モジュール間の信号線の制御が簡単にはおこなえないためである。テスト回路の追加の結果、既存のコアのテストベクタ資産をそのまま使用することができるようになった。

4．M32／ASSPの論理検証、テストベクタ生成法

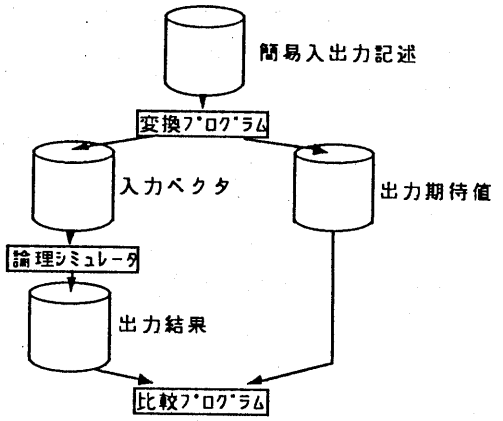
設計と同様、論理検証についても機能モジュール毎に行うことにした。モジュール毎の検証では、論理シミュレーションの入力ベクタの作成に、自作のバスアクセス指向の入出力信号記述（以下「バス指向信号記述」と略す）を用いて効率化を図った。また、チップ全体で検証すべき項目を限定した上で、全体での論理シミュレーションを実施する際にはMPUコアの部分をハードウェア記述で表わし、シミュレーション時間の短縮を図った。さらに、論理シミュレーション結果からテストベクタを生成する際に、モジュール毎の論理検証の資産を最大限利用することにした。そのため検証対象のモジュール以外の論理を簡単な回路でモデル化し、チップからの全ての入出力信号を生成した。

1）バス指向信号記述の利用

論理シミュレータの入力には、全ての入力信号値の記述が要求される。人間が個々の信号線の変化を記述していてはミスが避けられない。新規に設計した機能モジュールはバスに接続されるものがほとんどであるため、バスを介した読み書きが簡単に記述できればミスが減ることが期待できる。

対策として、入力ベクタおよび出力期待値は、自作のバス指向信号記述を自作のプログラムにより変換し生成することとした。バス指向信号記述はバス動作を簡潔に表現するとともに他の信号線の個々の動作を記述できるものにした。変換プログラムは、UNIXのツールとC言語で書いたプログラムを組み合わせて作成した。第2図に論理シミュレーションのフローを、第3図にバス指向信号記述の例を示す。

第3図ではDMAコントローラの例を示したが、異なるモジュールに対しても、レジスタの名前と数本の専用入出力線の入れ替えがあるだけである。



第 2 図

```
(時刻)  (信号記述)              (コメント)
  #     im8ch                  (パターン名)
  0     init                   (初期設定)
  8     write bcr0 6080        (ﾚｼﾞｽﾀ書込み)
   :                                :
 50     req x0xx xxxx          (転送要求信号)
 53     bus xx 0               (ｱｰﾋﾞﾄﾚｰｼｮﾝ)
 58     mread 111 12345678     (DMA転送)
   :                                :
 92     read bcr1 0FFFFFFF     (ﾚｼﾞｽﾀ読出し)
   :
```

第 3 図

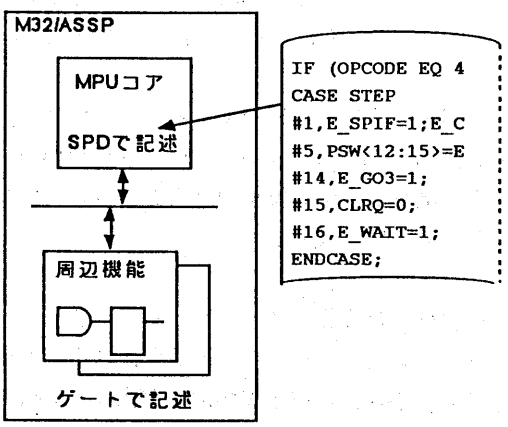
このバス指向信号記述を用いた結果、単純な記述ミスを減らせただけでなく、入力ベクタおよび出力期待値作成の時間を大幅に短縮し、さらに見やすく修正の容易な記述を得ることができた。

2）ハードウェア記述の利用

チップ全体のモジュール相互間の検証では、論理量が大量になるためシミュレーション時間の増大が問題となる。今回の開発では、機能モジュールの論理検証はモジュール単体でおこなったため、チップ全体で検証する必要のあるのは、バスを介したモジュール間のレジスタの読み書きと、数本

の専用信号線の動作だけに限定できた。

このように検証すべき項目を限定した上で、最も論理量の多いＭＰＵコアについては、モジュール相互間の検証に必要な機能のみをハードウェア記述で表わすことにより、論理シミュレーション時の論理量を減らし、効率の向上を図った。（第４図）　実現した機能は、転送命令、割込みからの復帰命令を含む４命令と、リセット、外部割込み、およびアービトレーション動作である。



第　４　図

ハードウェア記述としては、当社開発のミックスドレベル論理シミュレータ(3)用の機能記述であるＳＰＤ（Simulation Primitive Discreption）を使用した。この記述は元来ゲートレベルの動作を記述する比較的低い階層の記述であるが、ＭＰＵコアの動作記述にも耐えうるものであった。

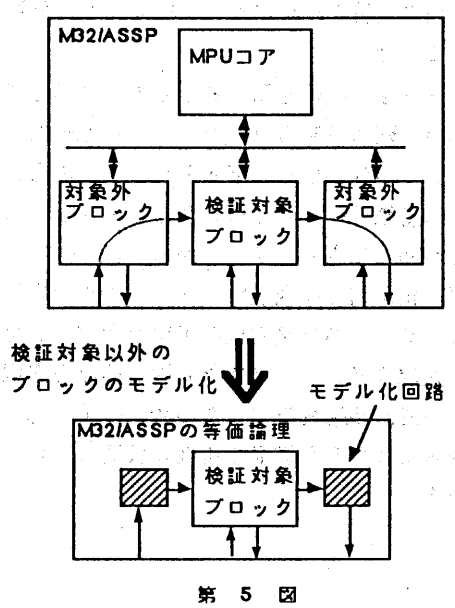
ハードウェア記述を用いてＭＰＵコアの動作を表わした結果、論理シミュレーション時間を大幅に短縮することが可能になった。

３）モデル化回路の利用

チップ全体の接続検証の論理シミュレーションからは、そのままチップ全体のテストベクタが生成できる。一方、論理検証を個別におこなう個々の機能モジュールについても、テストベクタは当然チップ全体を対象としたものでなくてはならない。モジュール毎の論理シミュレーション入力をテストベクタ生成に活かすために次の方法を取った。

まず機能モジュールのテストは１モジュールずつおこなうこととし、それぞれのテストの際、対象外のモジュールは停止状態に設定することにした。またテスト対象モジュールの出力がテスト対象外のモジュールを経由してチップ外に出力される箇所では、テスト対象外のモジュールを適当なモードに設定することによりチップ外部からの観測を可能にした。設定に必要な電源投入後あるいはリセット直後の入力シーケンスと、このシーケンスによって固定化されたテスト対象外モジュールの動作を表現する簡単なモデル化回路とを、トップダウン的に決めた。こうして、機能モジュールの設計担当者が、決められたモデル化回路を担当した回路に付加し、決められた初期設定シーケンスをモジュール毎の入力ベクタに付加して論理シミュレーションをおこなうことで、チップ全体に対するテストベクタを生成した。（第５図）

上記モデル化回路と初期設定シーケンスを使用した結果、テストベクタ生成はほとんどが機能モジュール単位でおこなえるようになり、論理シミュレーションの時間が短縮された。



第　５　図

ＭＰＵコアの部分については、先に述べたよう

にテスト用のモードを設け、チップ外部から全ての入力を与え、全ての出力を観測できるようにした。そのため既にあるＭＰＵコアのテストベクタをそっくりそのまま利用することができた。

5．まとめ

1）　ＴＲＯＮ仕様に基づく３２ビットＭＰＵコアを内蔵した機能集積型マイクロプロセッサを開発した。現在、常温で２５ＭＨｚまでの動作を確認している。

2）　ＭＰＵコアは既設計のレイアウトデータを用い、周辺機能の設計にはスタンダードセル手法を用いて工期の短縮を図った。機能のモジュール化に加え、テスト回路の利用によるテストベクタのモジュール化を進めた。

3）　バス指向信号記述、ハードウェア記述、モデル化回路を利用して論理シミュレーションを実施し、効率のよい検証とテストベクタ生成をおこなった。

バス指向信号記述を用いたことで、論理シミュレーション用の入力ベクタと出力期待値とを見やすい形で一括管理することができ、またベクタ生成の工期の短縮と単純なベクタ記述ミスの低減とが達成できた。

ハードウェア記述を用いることで、分割して設計を進める際に分割されたモジュールと外部との接続部分の検証が早い時期からおこなえる点、全体の論理接続検証時に論理シミュレーション時間の短縮が図れる点でメリットがある。

モデル化回路と初期設定シーケンスを使用したことで、テストベクタ生成の工期を短縮できた。

4）　大規模ＭＰＵをコアとし周辺機能を内蔵するための設計手法を確立したので、今後、応用分野を絞ったＡＳＳＰ製品を迅速に展開していく予定である。

謝辞

最後に、本製品の開発に助言いただいたＣＡＤ部門およびＬＳＩ研究所の関係各位に感謝致します。

参考文献

(1) K.Sakamura,"Architecture of the TRON VLSI CPU",IEEE MICRO April 1987,pp17-31.

(2) 吉田他　「ＴＲＯＮ仕様マイクロプロセッサＧＭＩＣＲＯ／１００のパイプライン処理構造」、電子情報通信学会集積回路研究会 1988-3-25 CPSY87-52,pp.25-30

(3) 田中他「ＭＯＳスイッチレベルシミュレーションにおけるアートワーク検証向き素子モデリング手法」、電子情報通信学会 CAS85-163-179,pp.49-56