オブジェクト指向手法をハードウェア設計に導入する提案 と VSI 向きコアのモジュール化

正員 中村 次男 (国際短期大学)

正員 笠原 宏 (東京電機大学)

A Proposal of Introducing the Object-Oriented Technique
for Hardware Design and the Modularizing of VSI Core

Tsugio Nakamura, Member(Kokusai Junior College)
Hiroshi Kasahara, Member(Tokyo Denki University)

It has become possible to integrate many circuit blocks as a system on chip. However it is difficult to design all circuit blocks from the beginning. On such trends, VSI is being expected as a new distributing commodities of the circulation, and as a reusable core for the designing of a large scale ASIC. But there are some serious problems for the standardization, and ease of use of the core.

In this paper, we propose a method of hardware modularizing which perform just as the data processing device in an object of the object-oriented technique. The user doesn't need to know the contents of an abstracted part called VC. It can get a result by sending a message with some parameters if we want to use the module. Therefore, it doesn't need to prepare for any programs and circuits newly. The hardware design method proposed here with the object-oriented technique is just for the VSI.

キーワード: VSI, VC, モジュール化, システムオンチップ, システム LSI

1. まえがき

集積回路技術の向上により、より多くの回路ブロックをシングルチップ上に実現できるようになってきた。プリント基板上に構築するシステムボードからシステムオンチップ (System On Chip: SOC) への移行が加速され、その応用範囲はコンピュータ、ファックス、携帯電話および多目的機器にと広範囲に及ぶ⁽¹⁾。このような大規模な機能ブロックをひとつのチップ上に実現可能となってきたが、すべてのブロックを最初から設計することは困難である。そこで大規模マクロセル (コア、メガセル) の標準化を図り、開発元の異なるそれぞれのコアを自由に選択して、ひとつのチップ上に集積できるというコアの新しい流通形態が注目されている。このような状況の下で VSI (Virtual Socket Interface) アライアンスが 1996 年に設立され、大規模 ASIC のコア流通が促進されている^{(2) ~ (4)}。

しかし、システムオンチップ (以下、SOC という) の現状には次のような課題が指摘されている⁽¹⁾。

(1) 益々高集積化する SOC に対して、求められる短期間

での製品開発。

(2) 多様な VC (Virtual Component) の設計に関する専門知識を ASIC 提供側は通常持っていない。

(3) 開発元の異なる VC の再利用法

以上の解決策としては、過去に開発された機能ブロックの再利用が挙げられるが、多様な環境下で開発された VC のインタフェースを標準化するのは知的財産権 (Intellectual Property: IP) も含めて、容易ではない。

本論文では、あたかもオブジェクト指向技術でのオブジェクト内のデータ処理装置のように働かせられる機能を持ったハードウェアのモジュール化法を提案する。抽象物である VC として部品化された中身を使用者は知る必要がなく、また部品内が他のモジュール動作によって破壊 (変更) されない。このように機能ごとの部品化を図り、内部を隠蔽してカプセル化できれば VSI 向きであるといえる。オブジェクト指向技術は再利用性や拡張性などのメリットから普及してきている。まず、用途が特定されない VC のモジュール化のようなハードウェア設計にオブジェクト指向技術を適用する (以下、オブジェクト指向ハードウ

ウェアという)意義について述べる。次に提案する設計法について述べ、その適用例として情報処理には不可欠な演算器を取り挙げる。中でも可変精度、特に暗号システムに見られるような近年の超多精度演算にも柔軟に対応可能な乗除算機構のうち、主に除算機構のチップスライズ化から演算'オブジェクト'への転換法、そして RSA 公開鍵暗号処理'オブジェクト'と VC の IP 保護機構'オブジェクト'への拡張例について述べる。SOC が現実的なものとなってきた経緯からハードウェア設計の見直しが求められてきていて、多く議論されているところであるが、筆者らが提案するオブジェクト指向手法を取り入れたハードウェア設計法はその一手段としてのものであり、他にこのような研究報告はまだ見当たらない。

2. オブジェクト指向ハードウェア構成法の提案

〈2.1〉オブジェクト指向技術の適用意義 コアの再利用に際しては設計者が各コアの仕様を明確に把握し、設計者の EDA ツールでそれぞれのコアを扱えることが必要であり、これにはコアの標準化が不可欠である。それは、製造技術に依存しない設計データや EDA ベンダ間で統一されたコアのデータ形式の実現ということになるが、各ベンダ間でそれぞれ特定の技術を持って競合している現状を考慮すれば、その標準化は大変な難問である。しかし、その標準化が達成されてもバーチャルなソケットで接続されるものとして設計された VC の使用に際しては更に大きな技術的問題が存在する。複数の VC が接続された SOC を使うには全ての VC の使用法を理解している必要がある。それらは複数の異なった開発元から提供されたものであるため、その難度は極めて高くなるということは容易に推察できる。

そこで、ソフトウェアの再利用性や拡張性などのメリットから普及してきているオブジェクト指向技術⁽⁵⁾をハードウェアにも適用することを提案する。オブジェクト化することにより、処理データの規模に左右されないソフトウェアのような柔軟性を持ったコアが実現できる。この結果メッセージとパラメータに関することが主な標準化となり、様々な制約からなる項目から解放され、標準化がしやすくなる。図 1 に示すように、使用者は VC に必要なメッセージ(その VC の機能)とパラメータ(その VC に与えられた機能を実行するのに必要なデータ)を全 VC に送るとそのメッセージに該当した VC が動作し、続けて送られるパラメータを取り込んで、その VC が処理した結果を返す。SOC 内の各 VC には実装時にアドレスコードのようなビットパターンを割り振り、コード変換表を介すなどして、そのビットパターンを除算や暗号といったメッセージとして全 VC に送る。各 VC にはこのメッセージを判断する回路を付加しておき、コアの起動を制御する。各 VC 間のメッセージ通信が必要な VC に関してはいろいろなバス調停機構が考えられる。一例として、バス使用の終了した VC が次の VC にバスの使用权を渡す機構を図 1 に示す。これにより使用者は個々の VC の中身(処理手順)を知る必要はなく、統一した使用法で簡単になり、VSI のように用途が特定されない VC の標準化が図りやすくなる。これを実現できるのが正にオブジェクト指向ハードウェア設計法である。そのためには、VC 内での処理に必要なクロック数、モジュール数(演算器やメモリなど)および I/O 制御などの各種制御を判断する機能を VC 自体に持たせる必要がある。

次に、デジタル信号処理、RSA 公開鍵暗号システムおよびシステム LSI などに不可欠な算術演算機構を例に挙げて、その設計法を述べる。

図 1 オブジェクト指向を取り入れた VC から成る SOC

Fig.1. SOC which consists of VCs with the object-oriented technique.

＜2.2＞算術演算器への適用 VSIのように用途が特定されないVCにおいては、使用されるときに精度の制限を受けず、標準化が図りやすいような仕様であることが求められる。これらの対策として、チップスライスは任意精度対応の有効な手段であるといえる。しかし、近年の高精度化傾向を考慮すると、必要なチップ数の選定と接続およびクロックなどの制御法が問題となる。予め必要と思われるチップ(ユニット)数をシステム内部に用意しておくが、それでも不足する場合はシステム外部に設置されたモジュールまでも制御する必要が生じる。このような場合、ひとつの制御部から全モジュールを制御する中央制御形では制御信号の追増加および信頼性に関して問題がある。そこで、数モジュールから成る各ユニット内に、演算に必要なモジュール数とクロック数の制御機能を持たせる。これにより、新たに回路を用意したり変更することなく、必要とされるユニット数とユニット内モジュールおよび全クロック数を演算機構が判断し、結果を返すことが可能になる。図2に演算機構の概要を示す。

図のように全ユニットをカスケード接続し、更に各々のユニットはバスに接続された状態で演算機構を構成する。数モジュールをカスケード接続した構成の1ユニットはオブジェクト指向を取り入れてチップスライズ化したもので、使用に際しては演算を指示するメッセージ(加減乗除算)の次に、精度情報と演算データのパラメータを演算機構に送る。演算機構では全ユニットがバスを介して精度情報を入力して記憶し、続いて最下位ユニット内の最下位モジュールから順に、必要なユニット数とその場合の最上位ユニット内モジュールへと演算データを取り込む。このとき、必要なモジュール数の判断機構は図3に示すように、

下位ユニットから、モジュール内データラッチ信号のメッセージを受けて、精度情報と比較しながら自モジュール内データラッチ操作を繰り返し、更にモジュール数が不足する場合は、その上位ユニットにデータラッチ操作のメッセージを伝える。このようにして任意の精度に対応したモジュール数が動的に決められるのと同時に、最下位モジュールから順に演算データが取り込まれる。必要な演算データが取り込まれると演算機構内では演算精度にあったクロックを用いて演算が実行され、演算結果がバスを介して返送される。以上の制御は各ユニット内に記憶された精度情報によって判断されるが、減算と除算機構の場合、結果の差あるいは商と剰余の精度範囲は“0”～減数/除数データのビット数と不定になる。その判断は逆に、最上位モジュールから下位モジュールに向かって、“0”がどのモジュールまで続いているかを知らせる同様な情報伝搬機構によって実現できる。

3.オブジェクト指向2項演算ハードウェア設計法

情報処理において不可欠な算術演算機構をオブジェクトとして取り挙げる。

基本演算器について、ある演算精度の加算/減算器を1モジュールとし、それを数モジュール用意した加算/減算機構を次のように定義する。

- (1) モジュールのカスケード接続により、1モジュールの演算精度の整数倍に精度を拡張することが可能。
- (2) 演算精度情報により、必要なモジュール数とクロック数およびクロック周波数などを各モジュールが判断し、演算結果を返す。演算に必要な判断機構は中

図2 演算機構の概要

Fig.2. Outline of the operation mechanism.

図3 モジュール数の判断と演算データのラッチ機構

Fig.3. Structure of the autonomous adjusting of the number of the modules and the latch of the operation date.

央制御形とならないよう、各モジュール間のメッセージ通信の伝搬により行う自己判断機構とする。パラメータに精度情報を付加することにより、演算精度の制限を受けず、任意精度に対応可能となる。

基本となる加算／減算モジュールのブロック図を図4に示す。加減乗除演算を指示するメッセージをバスを介して入力し、加算／減算であればnビット加算器あるいは減算器に切り替える。続けて、演算精度をモジュール内に記憶し、演算に最適なクロック周波数を分周器を通して選定する。そして演算に必要なデータの取り込みとモジュール数の決定を図2と図3で示したように精度情報と比較しながら行う。そのための上位と下位モジュール伝搬用メッセージとしてデータラッチ信号を設ける。加減算器の接続は桁上げ／桁借りデータを下位桁から入力して、上位桁へ出力することにより、容易に実現できる。

演算クロック数の判断は、記憶してある精度情報と比較しながら分周器出力の演算クロック数をカウントし、必要なクロック数を制御する。多精度演算時は多くのモジュールのカスケード接続になる。従来の一系統クロックに同期した演算では多精度時のクロックスキューが問題になるが、本機構では各モジュール分のクロック用として考えれ

ばよく、精度増に対するクロックスキューは問題とならない。一つ前の演算結果を半クロック間ラッチしておくことにより、各モジュール間のクロックの位相差は半クロック以内であればよく、演算結果の更新（ラッチ）はその半クロック後であることによる。

減算と除算結果のビット数は最大、減数（あるいは除数）のビット数～0と不定になる。そこで演算結果が上位モジュールからどのモジュールまで“0”が続いているかを伝えるデージーチェーンを構成することにより、演算結果の精度を判断することができる。そのための信号が結果の精度判断信号（ARIとARO）である。図5に演算結果の精度判断機構を示す。デージーチェーン用の精度判断信号ARIとAROの関係は、ARIがLowでかつ自モジュールの演算結果が“0”のときAROをLow出力する。

$$\overline{ARO} = \overline{ARI} \cdot (\text{演算結果が“0”})$$

従って、図のように最上位モジュールのARIはプルダウンし、ARIがLowでAROをHigh出力しているモジュール以降の結果を有効桁数と判断することができる（図では左の最上位モジュールから3番目のモジュールが有効桁の最上位を示す）。

以上はチップスライス化が実現できることを前提にし

図4 加算／減算の基本モジュール
Fig.4. Basic module of the addition/subtraction

図5 演算結果の精度判断機構
Fig.5. Precision judgment mechanism of the operation result.

ている。次に、チップスライシ化が最も困難とされてきた除算器のチップスライシ化とオブジェクト指向手法を取り入れた除算機構の試作結果について報告する。

4. 順序回路形除算機構のチップスライシ化

筆者らはこれまでに、特に演算に多くの時間を要する乗除算に関し、演算精度に応じて演算機構を構成することによって効率的な演算を実行する分散処理アーキテクチャの開発を行ってきた^{(6)~(9)}。特に除算器はこれまで演算桁数を越えた使用が不可能であったが、数モジュールの除算器に除算データを振り分けて除算を分散処理するチップスライシ化の実現に成功した。その結果、演算精度に対し、ダイナミックに除算モジュールの結合が行われ、要求された任意の精度で除算実行を容易に行うことが可能となった。本方式は完全なハードウェア化で実現でき、ソフトウェアのような柔軟性と開発効率の問題が解決され、しかもチップスライシ化により低コストでの実現が期待できる。

並列形はクロックや制御回路を必要とせず高速演算向きであるが、多精度演算においては集積度とファンイン／ファンアウト増の問題が伴う。一方、順序回路形はクロックを用いる分、並列形よりは演算速度が低下するが構成ゲート数が少なく、多精度演算向きであるといえる。次に、筆者らが開発した順序回路形チップスライシ機構について述べる。

〈4.1〉チップスライシ機構 本除算器⁽⁸⁾のアルゴリズムは減算シフト方式に基づいたものである。本除算モジュールを図6に示す。

図6 本除算モジュール
Fig.6. Divider module.

自モジュール内 n ビットの除数 $Y_{n-1} \sim Y_0$ と被除数との減算は、被除数が1ビットシフトされるごとに行われ、 n 回(ステップ)繰り返されて1除算モジュール分の除算が終了する。このシフトされる被除数、減算結果および部分剰余の伝搬入出力をそれぞれ以下のような各記号で定義する。

データ伝搬の入出力

QI : 被除数のシフト入力
QO : 被除数のシフト出力
BI : 減算結果の桁借り入力
BO : 減算結果の桁借り出力
RI : 部分剰余のシフト入力
RO : 部分剰余のシフト出力

あるステップ時の被除数を $X_{n-1} \sim X_0$ 、その1ステップ前の部分剰余を $D_{kn-1} \sim D_{k0}$ および減算抑止信号を P としたとき、減算結果の差 $D_{n-1} \sim D_0$ は各減算時の桁借り出力を $B_{n-1} \sim B_0$ とすると以下の論理式で示される。

$$D_0 = RI \oplus Y_0 \oplus BI$$

$$D_1 = (P \cdot D_{k0} + \bar{P} \cdot X_{1-}) \oplus Y_1 \oplus B_0$$

.....

$$D_{n-2} = (P \cdot D_{kn-3} + \bar{P} \cdot X_{n-2}) \oplus Y_{n-2} \oplus B_{n-3}$$

$$D_{n-1} = (P \cdot D_{kn-2} + \bar{P} \cdot X_{n-1}) \oplus Y_{n-1} \oplus B_{n-2}$$

以上の $D_{n-1} \sim D_0$ が新部分剰余 $D_{kn-1} \sim D_{k0}$ になる。また、各減算結果の桁借り出力は以下の論理式で示される。

$$B_0 = (RI \oplus Y_0) \cdot BI + RI \cdot Y_0$$

$$B_1 = [(P \cdot D_{k0} + \bar{P} \cdot X_1) \oplus Y_1] \cdot B_0$$

$$+ (P \cdot D_{k0} + \bar{P} \cdot X_1) \cdot Y_1$$

.....

$$B_{n-2} = [(P \cdot D_{kn-3} + \bar{P} \cdot X_{n-2}) \oplus Y_{n-2}] \cdot B_{n-3}$$

$$+ (P \cdot D_{kn-3} + \bar{P} \cdot X_{n-2}) \cdot Y_{n-2}$$

$$B_{n-1} = [(P \cdot D_{kn-2} + \bar{P} \cdot X_{n-1}) \oplus Y_{n-1}] \cdot B_{n-2}$$

$$+ (P \cdot D_{kn-2} + \bar{P} \cdot X_{n-1}) \cdot Y_{n-1}$$

上式は、初段の入力 RI には被除数の MSB が1ビットシフトした状態での演算操作開始を意味している。つまり、最上位モジュールの被除数は MSB から最下位モジュールの RI に伝搬し、1除算操作終了時には RI に X_0 が与えられる。 D_{kn-1} と最終段の減算結果の D_{n-1} は減算抑止信号 P によって選択され、出力 RO から上位モジュールの入力 RI に自モジュールの部分剰余として伝搬する。

$$RI \leftarrow RO = P \cdot D_{n-1} + \bar{P} \cdot D_{kn-1}$$

また、モジュール内最上位桁借り B_{n-1} は出力 BO から上位モジュールの入力 BI に桁借り出力として伝搬する。最上位モジュールの B_{n-1} が減算抑止信号 P として、全モジュールにフィードバックされる(全モジュールを連結して減算した結果、負のとき P は“1”)と同時に商として QI

に伝搬する。

例えば、4ビット除算器の場合、除算データの被除数と除数および除算結果の商と剰余はそれぞれ4ビットで、以下のように各記号を定義する。

被除数 X3, X2, X1, X0
除数 Y3, Y2, Y1, Y0
商 Q3, Q2, Q1, Q0
剰余 R3, R2, R1, R0

1モジュールの各ステップを表1に示す。表1において、各ステップではBIも含めて除数Y3～Y0との減算を行い、その結果の桁借り出力がBOになる。除算のオーバーフロを避け、すべての演算を可能とするため、被除数の上位4ビットX7～X4は初期設定ですべて“0”にイニシャライズされる。従って、1モジュールの被除数は8ビットであるが、実際の演算は4ビット÷4ビットである。

表1からkステップにおけるi桁目の部分剰余Riは次式で示される。

$$R_i = X_{i \cdot k + 4} - Y_i - BI$$

各ステップにおいて最終桁借り出力が正（“0”も含む）で商は“1”（負で“0”）になり、Q3から順にQ0へと商が求まる。

表 1 1モジュールの除算ステップ
Table 1. Division steps of one module.

ステップ	カスケード出力			部分剰余				カスケード入力		
	RO	QO	BO	上位		下位		BI	RI	QI
1	R3	X3		X6 X5 X4 X3		BI-1	R-1	X-1		
			BO3	Y3 Y2 Y1 Y0						
2	R2	X2		X5 X4 X3 X2		BI-2	R-2	X-2		
			BO2	Y3 Y2 Y1 Y0						
3	R1	X1		X4 X3 X2 X1		BI-3	R-3	X-3		
			BO1	Y3 Y2 Y1 Y0						
4	R0	X0		X3 X2 X1 X0		BI-4	R-4	X-4		
			BO0	Y3 Y2 Y1 Y0						

[注] 負の添字-1～-4はそれぞれ1桁下位～4桁下位のビットを示す。

<4.2>分散処理アーキテクチャ 4.1のチップスライズ化理論によって、今まで不可能視されていた除算の分散処理が可能になり、nビット除算にはnクロックで効率的に除算を実行する分散処理アーキテクチャが実現できる。

図7は1除算器(モジュール)の精度を8ビットとして、m個用意し、8～8mビット精度の除算機構を示したものである。SW1, SW2, SW3……は除算精度情報によって、必要な除算器を接続する機構で、その概念をスイッチで示したものである。例えば、8ビット精度ではすべてのスイッチはオフ、16ビット精度ではSW1だけがオン、そして8mビット精度ではすべてのスイッチがオン状態になる。当然、被除数と除数は図の下位8ビットから8ビット単位

に、左側除算器から順次それぞれの除算器に与えられる。

実際に8ビット除算器をワンチップ化し、ボード上に4チップ実装し、8～32ビット除算の実験を行った。

次に、図7の除算機構を内蔵した任意精度除算ユニットの試作について報告する。

図 7 可変精度除算機構
Fig.7. Structure of the division expandable to any degree of accuracy.

5. チップスライズ化から演算オブジェクトへの転換

先に開発した除算器のチップスライズ化とボード上での試作（4.）は、予め用意しておいたシステム内部で処理できる程度の演算精度を想定し、従来のような中央制御形によるものであった。しかし、前述したような超高精度化対応やVSIのVCを考慮し、これまでの中央制御形ではなく、各演算チップがオブジェクト指向手法を取り入れ、演算精度に合わせて複数の同一演算チップだけの動的な接続で任意の精度に対応可能な機構を筆者らは新たに開発し、試作実験を行った。

基本的には3.で述べたように、カスケード接続した数モジュールをメッセージとパラメータに応じて制御する処理機構組込形のユニット構成になる（図8）。1ユニットの処理能力を超えたパラメータに対しては、更にカスケード接続したユニット間のメッセージ通信により必要なユニット数が用いられて、処理結果を返す。

次に、筆者らが提案するオブジェクト指向ハードウェア設計法について、試作を通してその動作確認を行ったので報告する。

<5.1>除算オブジェクトの概要 本除算オブジェクトは計算機（ホスト）の高速で任意精度対応補助除算機構としてのものである。ただ大きな障害としてはシステムへの実装の問題が挙げられる。優れた機能をPLDやFPGAなどでチップ化しても、例えば、計算機内に組み込んだり、システムバスに接続するにはハードウェアの知識は不可欠であり、だれにでも容易に行えるものではない。そのため、PCIバス接続用のインタフェース基板などが市販されているが、どの機種の計算機にも簡単に接続できる実装の容易性を考慮し、先ず演算速度よりもシステムの動作確認を目的に、RS-232C端子に直接接続するタイプの除算ユニ

図 8 演算オブジェクト機構
Fig.8. Structure of the arithmetic object.

ットを試作した。1ユニットは順序回路形8ビット除算器4個構成とした。従って、本除算ユニットはひとつで8から32ビットの可変精度除算を効率よく行い、更に同除算ユニットをカスケード接続することにより、任意の精度に対応可能である。

必要な可変長除算データをホスト側から入力し、精度に比例したクロック数で除算を実行後、結果をホスト側に返送する。本除算ユニットの構成図を図9に示す⁽¹⁰⁾。

RS-232C機能も含めた本除算ユニットのワンチップ化を想定し、RS-232Cインタフェースは必要最小限の機能に限定した。本除算ユニットを直接ホスト側のRS-232Cコネクタに差し込むことでホスト側と接続される。試作では動作確認を目的にRS-232Cインタフェースとしたが、SOCやシステム内部に実装する場合はバスインタフェースが適している。可変精度除算機構は8~32ビットの除算

を精度に比例したクロックを用いて実行する。ホスト側からの精度情報を付加した8~32ビットの除算データをバッファ機構に格納する。8除算データまで格納できるバッファとして試作したが、ホスト側からの送信要求がなくなった場合、バッファが満杯になると可変精度除算機構で精度に合わせて順次除算を実行し、除算結果をバッファに蓄えた後、精度情報を付加して全除算結果をホスト側に返送する。

図9内の除算機構は任意精度対応可能であるため、図9の除算ユニットを更にカスケード接続することによって、1ユニットの整数倍に精度を拡張することが可能である。1ユニットをFPGA(Xilinx社製, XC5210, 84ピンPLCC)にワンチップ化し、それを4チップカスケード接続して8から128ビットの可変精度除算を実験により検証した⁽¹¹⁾。

〈5.2〉RSA公開鍵暗号処理オブジェクトの概要 情報セキュリティ技術として公開鍵暗号システムの標準となっているRSA暗号に提案するオブジェクト指向手法を取り入れた乗除算機構を組み込んでワンチップ化し、試作実験を行った。RSA暗号では暗号化と復号において

$$A^m \equiv R \pmod{n}$$

の剰余演算を行う。容易に解読できないよう鍵のnの精度は現在では1,024ビットが必要とされ、コンピュータの性能向上により、更なる高精度化が進むと考えられている。

このような超高精度演算においては多くの演算時間を要するため、RSA暗号は主に秘密鍵の配送やデジタル署名といった短いデータ用に使われている。そこで、効率的なアルゴリズムあるいは専用ハードウェアでの高速化が重要課題となっており、実際にワンチップ化した例も報告されている^{(12)~(15)}。しかし、複雑な演算機構の分散処理化を実現するのは容易ではないため、チップスライスを考慮したものではなく、更なる精度増には再開発が必要となり、またワンチップに実装するには極めて高集積なものとなる。

筆者らが試作したRSA公開鍵暗号システム⁽¹⁶⁾の1チップは16ビットと32ビット乗除算器をそれぞれ4モジュール内蔵(乗算16~64ビット、除算32~128ビットの可変精度)で、制御回路も含めてゲート規模は約1万6千ゲート(CMOSゲートアレイ, 100ピンQFP)である。概

図 9 1 除算ユニットの構成図
Fig.9. Block diagram of one divider unit.

要を図 10 に示す。

試作したチップをボード上に 4 個実装し、16～256 ビット長の鍵に対する暗号化／復号の実験を行いチップスライズ化が正常に動作することを確認した。図 11 に示すように、実験用ボードは PCI バスインタフェースボード（GPIO-100MN）を介してホストコンピュータ（MMX Pentium, 233MHz）に接続される。例えば、暗号化ではホストコンピュータから、暗号に相当するメッセージに続いて、平文 A そして公開鍵 n, m のデータがパラメータとして順に実験用ボードに送られる。実験用ボードでは精度に合わせたべき乗剰余演算が実行され、結果の暗号文 R がホストコンピュータに返送される。

1,024 ビットのべき乗剰余演算時は 16 チップのカスケード接続になるが、これらの制御は完全ハードウェア化によって実行される。電子機器の性能向上により、要求される演算精度数も更に高くなるといわれているが、オブジェクト指向手法を取り入れることによりカスケード数の追加だけで容易に対応可能である。

＜5.3＞SOC への実装 図 1 の SOC に除算オブジェク

図 10 RSA 公開鍵暗号処理オブジェクト機構

Fig.10. Structure of RSA public key cryptosystem processor object.

図 11 RSA 公開鍵暗号システムの実験

Fig.11. The experiment of RSA public key cryptosystem.

ト（図 9）と RSA 公開鍵暗号オブジェクト（図 10）がそれぞれ複数個から成る VC を実装した例を図 12 に示す。バスの使用権を得た VC が平文を暗号化（復号）したい場合、メッセージとして暗号、パラメータとしては平文（復号時は暗号文）と公開鍵（復号時は秘密鍵も）をそれぞれ精度情報を付加して、バスに順次送出する。暗号メッセージを受けて、RSA 公開鍵暗号の VC が起動し、続けて送られるパラメータを取り込み、精度情報によって必要な数の同暗号オブジェクトが動作して、暗号文（復号文）をバスに送出する。この処理結果を暗号のメッセージを送った VC が取り込む。続けて除算を行う場合は、同様に、除算メッセージと除算データをバスに送出すれば、除算の VC が起動して、除算結果得ることができる。次に行う処理がなくなったら、バスの使用権を次の VC に渡す。

図 12 SOC への実装例

Fig.12. Example of the implementation for SOC.

＜5.4＞VC の IP 保護機構オブジェクト VSI においては、VC の不正コピーや使用を防止するため VC（コア（メガセル））の IP 保護が大きな課題となっている。そこで、RSA 公開鍵暗号システムの認証機能を VC の認証に利用した保護機構を図 13 に示す。

各コア・ベンダから与えられたアクセス権情報を製造時にチップ内の ROM に格納しておき、起動時（電源オン時）に RSA 公開鍵暗号の認証機能を使って各 VC に対して、それぞれアクセス権を与える。アクセス権情報とは VC の使用を認める暗号化署名文としてコア・ベンダから与えられたもので、VC 内に内蔵されているコア・ベンダ側の秘密鍵と使用者の公開鍵を用いて、剰余演算を繰り返して、署名文を復号し、使用者を確認する。この動作は起動時だけ、認証処理機構によって行われる。

ROM 内には各 VC の同様なアクセス権情報が記憶されており、順次アクセス権情報を全 VC が取り込み、自分のアクセス権情報であるか復号する。自分用アクセス権情報であれば、その VC を使用可能な状態にし、暗号化署名文の復号演算操作を終了する。最後の情報まで合致するものでなければ VC は非アクティブな状態のままで、その VC を使用することはできない。VC を使用可能にするかどうかは幾つかの機能ブロックのストローブ信号により制御

図 13 VC の IP 保護機構

Fig.13. Structure of the protected IP of VC.

する。従って、各 VC 内ごとにストローブ信号は異なるものであり、全てのストローブ信号を解析し、不正使用することは困難である。また、ROM 内には複数の VC の長さの異なるアクセス権情報が記憶されており、どの VC 用であるか解読するのは簡単ではない。

各 VC に内蔵された乗除算機構は一般の演算用としても用いられる。この場合は数モジュールのカスケード接続から成る乗算および除算機構は乗算および除算処理機構によって演算精度に合ったモジュール数の選定や演算に必要なクロックを用いて、結果を返す。従って、暗号システムのみとして取り除くことはできず、IP の保護機構は強固なものとなる。

6. むすび

システムオンチップと呼ばれるような集積回路の大規模化が実現可能となってきたが、すべての回路ブロックを最初から設計することは困難な状況になってきている。そこで、大規模 ASIC の設計に使うコアの流通・再利用という新しい流通形態として VSI が期待されている。しかし、コアの標準化や IP などの大きな問題点が存在し、コアの使用に際してはその使用容易性が求められる。

本論文ではソフトウェアのような柔軟性を持ったコアとして標準化し易いモジュール化技術について、先ずオブジェクト指向手法を取り入れたハードウェアの適用意義と設計法について述べ、次にこれまで分散処理が困難であった除算器を取り挙げて、オブジェクト指向による任意精度除算システムおよび情報セキュリティ技術としての RSA 公開鍵暗号処理オブジェクトと VC の IP 保護機構オブジェクトへの拡張例について報告した。

メモリチップ上に演算素子を集積したシステム LSI が次世代の半導体生産をめぐる新たな試みとして注目されており、今後その需要は急増されると見られている。通常、演算データはバスを介して CPU に送られ、CPU での演算結果をまたバスを介してメモリに格納という手順をふむのに対し、メモリチップ内で演算処理を可能にするもの

である。その場合、演算に必要なと思われるモジュール数をあらかじめシステム内に（不足する場合は外付けで）用意しておく。演算が必要ならばこのオブジェクト指向ハードウェアの演算機構へ、メッセージ（加算、減算、乗算、除算）をパラメータ（精度情報および演算データ）とともに送ることにより、結果のみを必要な精度で得ることができる。この場合、演算時に必要な演算モジュール数の選定、演算データの分割および演算クロックの制御などは演算機構内で行うので、新たにプログラムや回路を用意する必要はない。本論文で提案したオブジェクト指向手法を取り入れたハードウェア設計法は正に VSI 向きであるといえる。

（平成 12 年 2 月 24 日受付、平成 12 年 9 月 11 日再受付）

文 献

- (1) M.Bimbaum & H.Sachs : "How VSIA Answers the SOC Dilemma", IEEE.COMPUTER, pp.42-50(1999-6)
- (2) 東北地区・大規模集積システム設計教育研究センタ、東北大サブセンタ、<http://michinoku.vlsi.ecei.tohoku.ac.jp/expD/auto-design/5/>
- (3) 日経エレクトロニクス：「ASIC のコア流通を促進する標準化団体を日米欧亜で設立」、<http://ne.nikkeibp.co.jp/NE/1996/960923/report1.htm>
- (4) 伊藤元昭：「第 2 フェーズに入った VSI、現実的なコアの流通方法を模索」、NIKKEI ELECTRONICS, No.701, pp.43-47, (1997-10)
- (5) 早瀬健夫：「オブジェクト指向フレームワークと製品特化 CASE による組み込みシステム開発」、電学論(C), Vol.119-C, No.12, pp.1573-1581(1999-12)
- (6) 中村次男、笠原宏：「任意精度の演算が可能な並列除算器のアーキテクチャ」、信学論(C-II), Vol.J75-C-II, No.5, pp.185-192(1992-5)
- (7) 中村次男、笠原宏：「任意の精度に被除数を拡張できる並列除算器の提案」、電学論(C), Vol.111-C, No.7, pp.273-278(1991-7)

- (8) 中村次男, 笠原宏: 「任意の精度に拡張容易な除算器の提案」, 電学論 (C), Vol.111-C, No.3, pp.123-128 (1991-3)
- (9) 中村次男, 笠原宏: 「拡張容易な乗算器モジュールとそのワンチップ化の提案」, 電学論 (C), Vol.110-C, No.2, pp.95-100 (1990-2)
- (10) 遊佐敦史, 笠原宏, 中村次男: 「RS-232C で接続可能な任意精度補助除算ユニットの試作」, 信学総全大, A-51 (1995-3)
- (11) 大石博朗, 杉山寿男, 鈴木領, 笠原宏, 中村次男: 「精度にあわせて効率的な除算を行う補助ユニットの開発」, 信学総全大, C-12-42 (1997)
- (12) 森田光: 「暗号技術と高速算法」, 情報処理, Vol.34, No.3, pp.336-342 (1993-3)
- (13) Brickel, E.F.: "A Survey of Hardware Implementations of RSA", Advances in Cryptology-CRYPTO'89, Lecture Notes in Computer Science 435, pp.368-370, Springer-Verlag (1990)
- (14) 石井晋司, 大山勝一, 山中喜義: 「高速公開鍵暗号プロセッサ」, 信学論, D-I, Vol.J80-D-I, No.8, pp.725-735 (1997-8)
- (15) 日本 IBM (株): 「1024ビット RSA 暗号アクセラレータ」, NIKKEI BYTE (1999-2)
- (16) 中村次男, 大石博朗, 笠原宏: 「RSA 公開鍵暗号システム実装におけるビットスライス化の一方式」, 電学論 (C), Vol.118-C, No.7/8, pp.1073-1081 (1998-7/8)

中村 次男 (正員) 昭 46 東京電機大学工学部電気工学科卒。平 2 同大学大学院修士課程了。昭 46 日本電気精器株式会社入社。現在, 国際短期大学情報通信科助教授。博士(工学)。デジタル演算器, 高速暗号化に関する研究に従事。著書「デジタル回路の基礎」(日本理工出版会) 他。IEEE 会員。

笠原 宏 (正員) 昭 45 東京電機大学博士課程満期退学。同大学助手, 講師, 助教授を経て現在, 工学部電気工学科教授。工学博士。パワーエレクトロニクス, 計算機システム, 制御用分散処理システム, オブジェクト指向システム設計法, インタラクティブにとらえた脈診情報処理方式, 高速暗号化に関する研究に従事。情報処理学会, 電子情報通信学会, IEEE, ACM 各会員。