■10 群（集積回路）－ 5 編（演算・信号処理 LSI）

2 章 実現アーキテクチャ

（執筆者：天野文雄）［2010 年 3 月 受領］

■概要■

本章では，信号処理／通信処理を実現するためのプラットフォームとして，CPU，DSP，アクセラレータ，ASIP とコンフィギュラブルプロセッサ，および FPGA アーキテクチャについて説明している．また，具体的な例として 6 社のプラットフォームを紹介している．

【本章の構成】

2-1 節では，CPU について発展の歴史，情報家電向けマイクロプロセッサの最新の動向，演算レベルでの並列化である SIMD 演算器や，命令レベルでの並列化であるスーパースカラ，VLIW，ハードウェアによるマルチスレッディングなどマイクロプロセッサ高性能化の技術動向，更にマルチコア化，メニーコア化の現状と情報家電向けの応用について述べている．

2-2 節では，ディジタル信号処理プロセッサ（DSP）の歴史，一般的な構成，VLIW とスーパースカラなどの高性能化技術の動向，マイクロプロセッサとの違い，ソフト開発ツール，およびプログラム開発手順の概要を述べている．

2-3 節では，プログラマビリティをもった専用ハードウェアエンジンとして，リコンフィギュラブルプロセッサと SIMD エンジンの例を紹介している．

2-4 節では，ASIP（Application Specific Instruction Set Processor）に関連して，コンフィギュラブルプロセッサの例，コンフィギュラブルプロセッサにおけるソフトウェアの再利用，検証などの問題点と今後，開発環境とその特徴などを紹介している．

2-5 節は，FPGA アーキテクチャの一般的な解説である．

最後に 2-6 プラットフォームでは，具体例としてパナソニックの Uniphier，ルネサステクノロジの EXREAL Platform，NEC エレクトロニクスの EMMA，東芝のメディア処理プラットフォーム，TI の DaVinci と OMAP，および富士通の組込み用途向けマルチコアプラットフォームについて解説している．

■10 群 － 5 編 － 2 章

2-1 CPU

（執筆者：山田哲也）［2008 年 12 月 受領］

2-1-1 マイクロプロセッサの登場（CISC と RISC）

図 2･1 にマイクロプロセッサの変遷を示す．マイクロプロセッサは 1971 年に Intel 社が開発した 4004[1] に始まる．ビジコン社の電卓向けの LSI の依頼を受けて，汎用デバイスとして開発したもので，4 ビットのアーキテクチャであった．

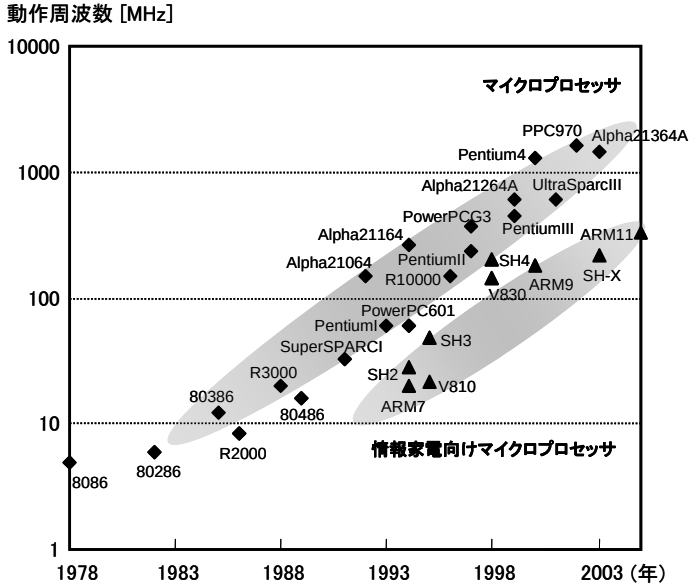


図 2･1　マイクロプロセッサの変遷

1970 年代から 1980 年代前半までは，文字データを扱えるようにデータ幅が 8 ビットに拡張され，Intel 社の 8008（1972 年），8080（1974 年）が開発された．その後，16 ビットのマイクロプロセッサ 8086（1978 年），8088（1978 年）が開発された．Intel 社の 8088 は IBM PC に，80286 は IBM PC/AT のマイクロプロセッサに採用された．その後，Intel 社は 32 ビットのアーキテクチャ（IA-32）の 386，486，Pentium I[2] ～III，Pentium 4，64 ビットアーキテクチャ（IA-64）の Itanium へと続いていく．

Intel 社のマイクロプロセッサ（x86 系列）のアーキテクチャは，プログラミングを容易にするために高度な機能をもつ命令セットやアドレッシングモードをもち，CISC（Complex Instruction Set Computer）と呼ばれる．CISC 型の製品として，ザイログ社の Z80（1976 年），Z8000（1979 年），モトローラ社（現 Freescale Semicondutor 社，以下 Freescale 社）の MC 6800（1974 年），MC 68000（1979 年）などがある．

1980 年代に，命令長を固定し，命令数の削減を行い，パイプラインを用いて実行速度を上げられるようにしたアーキテクチャとして RISC（Reduced Instruction Set Computer）が登場し

た．パイプラインとは，命令の処理を，命令フェッチ，デコード，実行，レジスタへの書き込みのように複数のステージに分け，複数の命令をオーバラップして実行する方式を指す．RISC では，ロードストア命令でメモリアクセスを行い，演算はレジスタ上で行う．パイプラインの停止（ストール）が起こると性能が低下するため，パイプラインの停止を避けるためにコンパイラでの最適化技術が重要となった．固定命令長の採用と命令セットの削減によるデコーダの単純化とパイプライン構成により，動作周波数を高くすることができる．スタンフォード大学，カリフォルニア大学バークレー校からそれぞれ MIPS，RISC I が提案され，これらの研究成果は MIPS 社の R3000（1988 年），R4000[3]（1992 年），Sun Microsystems 社の SuperSPARC I（1992 年），UltraSPARC I（1995 年）などに引き継がれた．

その後，各社でRISCプロセッサが開発され，多くの高速化技術が実装された．IBM社，モトローラ社（現Freescale社），Apple社の共同開発で，IBM社のPowerアーキテクチャをシングルチップ向けに変更したPowerPC 601[4]（1993 年），PowerPC 604[5]（1994 年），PowerPC G3～G5（1997～2003 年）や，HP社のPA-RISCアーキテクチャのPA-7000（1989 年），PA-7200[6][0]（1994 年），PA-8000（1996 年），DEC社（現HP社）のAlpha 21064[7]（1992 年），Alpha 21164[8]（1995 年），Alpha 21264[9]（1999 年）などがある．Intel社もPentium以降はハードウェア内部ではRISCプロセッサと同様の高速化技術を実装している．

2-1-2 情報家電向けマイクロプロセッサ

1990 年代になると，制御を中心としたマイコン業界でも，ディジタル情報機器向けに新たな RISC 型のマイクロプロセッサが開発された．民生機器の厳しいコスト要求と実装面積の省スペース化に応えるため，低コストと低消費電力を特長とする．ARM 社の ARM/Cortex ファミリー，日立（現ルネサステクノロジ）の SuperH ファミリー，NEC（現 NEC エレクトロニクス）の V800 シリーズをあげる．

ARM アーキテクチャは, Acorn Computer 社が開発した ARM 2, ARM 3 を受け継ぎ, 同社, Apple 社，VLSI Technology 社が出資した Advanced RISC Machines（ARM）が考案した 32 ビット RISC アーキテクチャである．分岐命令を効率よく実行するために条件コードを付加した命令フォーマットをもつ. Apple 社の PDA である Newton に採用された ARM6[10]（1991 年）はメモリ管理機能を搭載した．ARM 7（1993 年）は多くの携帯電話に採用されている．コードサイズを削減するために，ARM 7 TDMI では頻繁に使用される 32 ビットの命令を 16 ビットの命令長に圧縮した Thumb 命令セットが実装された．

その後，開発された ARM 9（1997 年），ARM 10（1998 年），ARM 11[11]（2002 年）では，ARM 7 のパイプライン段数の 3 段からそれぞれ 5 段，6 段，8 段とし周波数を向上させた．ARM 11 では分岐予測機能を内蔵した．

その後，モバイルなどの情報アプリケーション向けの Cortex-A，車載などのリアルタイム処理向けの Cortex-R，産業・民生機器制御向けの Cortex-M と応用用途ごとにアーキテクチャを分離した[12]（2004 年）．Cortex-A8[13] では，パイプラインを 13 ステージと細分化し，動作周波数を向上させた．Cortex-A9 では，パイプライン段数は 8 ステージで Cortex-A8 に対し高周波数型ではないものの，クロック当たりの性能（IPC：Instruction Per Cycle）を向上させるために最大 4 命令発行のアウトオブオーダのスーパースカラ技術を実装した．

SuperHアーキテクチャは，16 ビットの固定命令長で，SH-1[14]（1992 年）はROM，RAM，

DMAC，割込みコントローラを内蔵するシングルチップマイコンとして開発された．SH-2 [15)0]（1994 年）では外付けメモリを前提とし，キャッシュメモリとSDRAMコントローラを内蔵した．SH-3 [16)]（1995 年）では，仮想記憶の実現のためにメモリ管理機構（MMU）を内蔵した．MMUでは，TLB（Translation Look-aside Buffer）にアドレス変換情報をキャッシングすることで高速にアドレス変換を行う．

信号処理の強化のため，SH-2，SH-3 にそれぞれ DSP 機能を内蔵した SH-DSP（1996 年）と SH3-DSP（1999 年）がある．DSP 向けに積和演算用の乗算器と ALU，DSP 専用メモリ，命令リピート機能を有するが, 命令フェッチやアドレス計算などの機能を CPU と共有したため，DSP 部分は小面積で済むのが特徴である．SH-4 [17)]（1998 年）では 2 命令発行のインオーダのスーパースカラを採用し，グラフィック強化用に内積演算器を内蔵した．SH-X，SH-X2（2004 年，2005 年）では分岐予測機能をもちパイプライン段数を 7，8 段とし周波数向上を行った．

V800 アーキテクチャは，16 ビット並びに 32 ビットの混在命令長である．5 段パイプラインの V810（1992 年）が最初に製品化され，シングルチップマイコン分野に V850 [18)]（1995 年），マルチメディア向けに V830 [19)]（1995 年）が開発された．その後，V850 は展開され，ハイエンド向けには V850 E2（2005 年）は，パイプライン段数を 7 段に増加して周波数が向上し，2 命令発行のスーパースカラ技術で周波数当たりの処理性能が向上した．

2-1-3 マイクロプロセッサの高性能化技術

マイクロプロセッサの進化の過程で，性能向上のため演算レベルや命令レベルで並列実行する手法が多く開発された．また，周波数を上げるためにパイプライン段数を増加する手法がとられた．

演算レベルの並列化では, 一般の DSP に搭載されている加減算と乗算を行う積和演算器を始めとして, 一つの命令で複数の演算を並列実行する SIMD 演算器（Single Instruction Multiple Data）や，ベクタ内積演算を行う内積演算器が知られている．SIMD 演算器は，マルチメディア処理において，複数の画素や複数の音声データをパックして，一命令で演算することなどに用いられる．SIMD 演算は, Intel 社の MMX [20)] や SSE（Streaming SIMD Extensions）[21)], モトローラ社（現 Freescale 社）の Altivec [22)]，IBM 社の VMX，Cell プロセッサの PPE，MIPS 社の MDMX，ARM 社の Neon，パナソニック社の Uniphier の DPP など多くのプロセッサに搭載されている技術である．

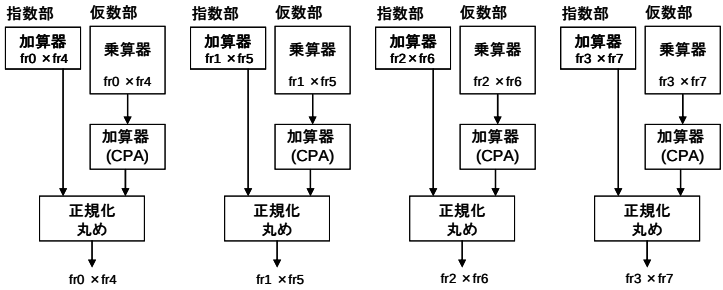


図 2・2　単精度浮動小数点 SIMD 乗算器の構成例

SIMD 演算器は，整数演算や浮動小数点演算に対応し，データ幅として 64 ビットや 128 ビットなどがあり，様々な算術論理演算を行うことができる．例えば，Intel 社の MMX では，データ幅 64 ビットの整数で，8 ビットは 8 個，16 ビットは 4 個，32 ビットは 2 個，64 ビットは 1 個のデータがパックされ，算術演算，論理演算，シフト演算などを同時に演算することができる．

図 2･2 に単精度浮動小数点 SIMD 乗算器の構成例を示す．パックされた 4 個の 32 ビット単精度データに対し，同時に演算を行う．4 演算を同時に実行するために，加算器，乗算器，CPA（Carry Propagation Adder），正規化回路，丸め回路は，それぞれ四つ搭載されている．

内積演算器は，SH-4 で搭載された演算器 [23] で二つの 4×1 ベクトルの内積演算を行う．グラフィックのアフィン変換のようなメディア処理の高速化向けで，用途が限定されるものの SIMD 演算器より少ないハードウェアで実現できる利点をもつ．**図 2･3** に単精度浮動小数点内積演算器を示す．乗算器と加算器は，入力数だけもつが，4-2 コンプレッサ（4 入力 2 出力の加算器）や正規化回路，丸め回路を一つにすることで論理規模が小さい．

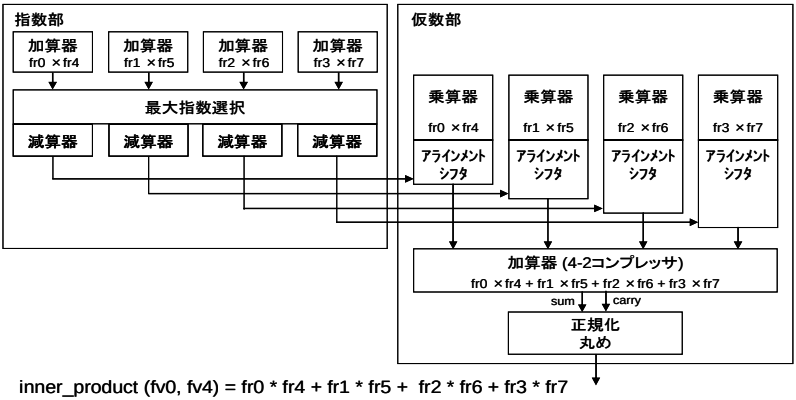


図 2・3　単精度浮動小数点内積演算器の構成例

次に，命令レベルの並列化では，1 クロックで複数の命令を実行するスーパースカラ，VLIW（Very Long Instruction Word），ハードウェアによる同時マルチスレッディングがある．スーパースカラは，命令間の依存関係を解決するハードウェアをもつが，VLIW はコンパイラで依存関係を解決することを前提に，このハードウェアをもたないため，構造を簡略化できる．スーパースカラは，命令の並びと同じ順番で実行するインオーダ型と，命令の並びによらず，実行できる命令から実行するアウトオブオーダ型がある．スーパースカラは，ほとんどの高性能マイクロプロセッサで採用されている技術である．アウトオブオーダ型はハードウェアが複雑なため，小面積のマイクロプロセッサではインオーダ型が用いられることが多い．VLIW 技術を採用した製品として，Intel 社の Itanium，Transmeta 社の Crusoe，富士通の FR-V などがある．アウトオブオーダ型スーパースカラの製品として，IBM 社の PowerPC 604，DEC 社（現 HP 社）の Alpha 21164（1995 年），Sun 社の UltraSPARC I（1995 年）などがある．

スーパースカラや VLIW において，命令間の依存関係には並列性の抽出に限界があり，並

列度は高々4 程度といわれている．そこで，一つのプログラムを複数のスレッドと呼ぶ処理単位に分割し，複数の独立したスレッドを並行処理する方法がとられるようになった．そのうちの一つが，ハードウェアによる同時マルチスレッディング（SMT：Simultaneous Multi-Threading）である．ハードウェアによる同時マルチスレッディングは，独立したスレッドでは，あるスレッドがキャッシュミスなどで待っている間も他スレッドは実行できるという点でメモリレイテンシの隠蔽に効果がある．ハードウェアでは複数のプログラムカウンタと複数のレジスタを有し，メモリ，キャッシュや演算器を共有する．製品としては Intel 社の Pentium 4 のハイパースレッディング技術 [24]，IBM 社の Power 5 [25] などがある．

周波数向上のため，パイプラインに対し処理の分割単位を細かくし，ステージ数を増加する手法がとられた．各ステージでの処理内容が少なくなり，各ステージの実行時間が短くなるため，周波数を向上させることができる．特に，ステージ数の多いパイプライン構造をスーパーパイプラインと呼ぶ．パイプラインステージ数を増加すると，各ステージでの依存関係のあるデータによるパイプラインの停止が発生しやすくなるため，レジスタ書き込みより前に後続命令に結果を伝えるフォワーディングを行う．フォワーディングとはデータが準備できた段階，例えばデータキャッシュからのロードや演算器の出力結果をレジスタに書き込むより前に，そのデータを使用可能とする技術である．

プログラム中に分岐命令がある場合，分岐先が実行される場合とされない場合で，プログラムの流れが変わる．分岐先が実行されるときにフェッチが待たされることで，パイプラインの効果が低下し，パイプラインステージが長いほどその影響が大きくなる．そこで，条件分岐に対し，過去の履歴を用いて分岐するかどうかを予測する分岐予測，あるいは分岐先アドレスを予測する分岐先予測などを用いて，分岐先の命令を予めフェッチしておくことで，分岐によるペナルティを削減するための様々な技術が開発されている．Intel 社の Pentium 4 は周波数向上を優先し，パイプライン段数は 20 段（のち 31 段に拡張）と多い．このように長いパイプラインでは分岐予測の性能は特に重要となる．

2-1-4 マルチコア，メニィコア

これまで微細化によりマイクロプロセッサの集積度と性能の向上を図ってきた．微細化に関して，Intel 社創業者の Gordon Moore 氏が提唱した「集積回路上のトランジスタ数は 18 か月ごとに倍になる」というムーアの法則が知られている．ムーアの法則では，18 か月ごとに CMOS プロセス技術は一世代シュリンクし，その度に，同じダイサイズに搭載できるトランジスタ数は 2 倍に増える．これは，スケーリング係数 k（約 1/0.7）を用いて表すスケーリング則では，プロセスが一世代微細化すると，駆動電圧が $1/k$（0.7 倍），ダイサイズ $1/k^2$（0.5 倍），ゲート酸化膜厚 $1/k$（0.7 倍），消費電力 $1/k^2$（0.5 倍），動作周波数 k（1.4 倍）となることを示しており，微細化による恩恵が多かった．

ところが，ディープサブミクロンプロセス（130 nm 以降）では，微細化による従来スケーリング則が効かなくなり，単体マイクロプロセッサの周波数向上が困難となってきた．周波数向上のために，電源電圧の低下とリーク電力増加対策としての高しきい値電圧化による回路動作速度低下のためである．

プロセッサ単体の周波数向上が困難なことに対処する技術がマルチコアである．マルチコアは，同じプロセッサ（CPU コア）で構成されたホモジニアス型と，異なる CPU コアで構

成されたヘテロジニアス型がある．ホモジニアス型は同じ CPU コアを用いるため，従来の単一のプロセッサの性能向上に用いられる．ヘテロジニアス型は，汎用の CPU コアと特定用途の CPU コアなど異なる CPU を組み合わせることにより，処理能力の効率向上を図ることや，従来の複数チップを単一チップに集積するときなどに用いられる．

マルチコアで用いられる並列処理方式の種類として，対称型マルチプロセッシング（SMP：Symmetric Multi-processing）方式と，非対称型マルチプロセッシング（AMP：Asymmetric Multi-processing）方式がある．SMP 方式とは，すべての CPU に対称に処理が割り付けられたマルチコアの並列処理方式のことで，AMP 方式とは，各 CPU に非対称に処理が割り付けられたマルチコアの並列処理方式のことである．SMP 方式では，個々の CPU コアがメモリ空間を共有するスレッドを実行し，CPU 間でキャッシュを共有するため，ハードウェアによるキャッシュのデータ一貫性保持機構が必要となる．AMP では，各 CPU コア上に OS とタスクが静的に割り当てられ，異なるプログラムを実行するため，ハードウェアによるキャッシュのデータ一貫性保持機構は不要である．リアルタイム性が必要な処理では，特定の CPU コアにその処理を割り当てることができるため，AMP 方式が適する．

マルチコアに関しては，次にあげる二つの法則を考慮する必要がある．

一つは，Intel 社の Fred Pollack 氏が提唱した「プロセッサの性能はそのダイサイズの平方根に比例する」という経験則のポラックの法則である．ダイサイズが 2 倍のとき，性能は $\sqrt{2}$（≒1.4）倍しかならないことを示している．ダイサイズ半分の CPU コア二つと CPU コアの性能を比較すると，ダイサイズ半分の CPU コアは，性能は $1/\sqrt{2}$ 倍だが，二つの CPU コアにより $2\times 1/\sqrt{2} = \sqrt{2}$ 倍の性能となる．簡単のため電力は同一周波数，同一電圧ではダイサイズに比例すると仮定すると，同じ電力で $\sqrt{2}$ 倍の性能を得られることになる．ただし，これは複数のコアで依存関係のないスレッドまたはプログラムを並列実行できる場合となる．

もう一つは，Gene Amdahl 氏が提唱した「プログラムの中の並列化できない部分が並列化による性能向上を制限する」のが，アムダールの法則であり，次の式で表される．

$$T_{\mathrm{multi}} = T_{\mathrm{single}} \times ((1-P) + P/S)$$

ここで，T_{multi}：マルチプロセッサの処理時間，T_{single}：単一プロセッサの処理時間，P：並列化可能な部分の比率（$0 \leqq P \leqq 1$），S：並列度（≒プロセッサ数）

アムダールの法則によると，例えば並列化不可能な部分が 20 %あると，プロセッサ数をいくら増やしても性能は高々5 倍にしかならない．

ポラックの法則を用いてマルチコア向けには単純かつ面積の小さいプロセッサを使用する傾向がある．

ソニー，SCE，東芝，IBM 社の共同開発の Cell[26]（2005 年）では，一つの汎用プロセッサコア（PPE）と 8 個の演算用プロセッサコア（SPE）を内蔵するヘテロジニアス型構成である．PPE は 2 スレッドの 64 ビット POWER アーキテクチャの CPU であり，OS や SPE へのリソース管理を行う．SPE は浮動小数点 SIMD 演算器をもち，単一の SPE で単精度浮動小数点演算を 4 演算同時に実行できる．キャッシュメモリをもたない代わりに，各 SPE に 256 kB の SRAM が搭載され，DMA（Direct Memory Access）ユニットを用いてメインメモリとのデータ転送を行う．演算性能が高く，ゲーム機 Playstation 3 をはじめ，サーバ，ワークステーション，スーパコンピュータにも採用されている．

Sun Microsystems 社は，サーバ用途として多数のスレッドが並行実行できる特性を利用し，Niagara プロセッサ [27]（2006 年）では 4 ウェイのハードウェアによる同時マルチスレッディングを行う CPU コアを 8 個搭載し，32 スレッドの実行を可能としている．

Intel 社は, PC 向けには, ハードウェアによる同時マルチスレッディングを行う高性能 CPU を 2 から 4 個搭載するチップの開発に加え, ポラックの法則を利用し, シンプルな構造な CPU を多数並べたメニィコアを提案している．将来のアプリケーションでは，アムダールの法則の制約である逐次箇所に比べ，並列化可能な箇所が増えるため，メニィコアの用途が広がるとしている．CPU コアを 80 個搭載した TILE 型のチップ [28]（2007 年）を発表している．各プロセッサは VLIW 型の命令セットをもち二つの浮動小数点積和演算器を内蔵し，2 次元のアレイ構造である．

2-1-5 情報家電向けのマルチコア

情報家電向けのプロセッサにおいてもアプリケーションの処理性能要求を満たすため，マルチコアのアーキテクチャが普及し始めている. 三つの ARM 926 コアと DSP を集積した NEC エレクトロニクス社の MP 211 [29]，四つの SH-X3 CPU コアを集積したルネサステクノロジ社の RP-1 [30]，汎用 CPU コア（IPP）と SIMD 型のマルチメディアアクセラレータコア（DPP）を集積したパナソニック社の Uniphier [31]，ARM 社の MPCore など様々なマルチコアが開発されている．

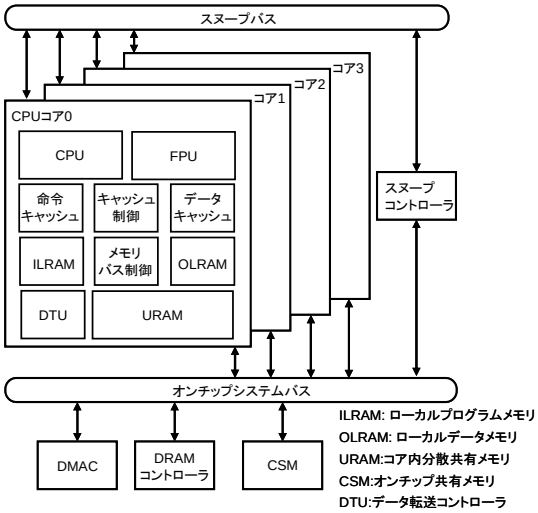


図 2・4　マルチコアの例（RP-1）

図 2･4 にマルチコアの例として，RP-1 のマルチコアと内蔵メモリのブロック図を示す．四つの CPU コアとスヌープコントローラがオンチップシステムバスとスヌープバスに接続されている．AMP と SMP の両方式をサポートし，4 CPU コア内で混在も可能である．SMP のデータキャッシュの一貫性保持は, スヌープバスを用いてスヌープコントローラが制御する．

複数の階層の内蔵メモリをもち，内蔵メモリのデータの転送には，DMAC とコア内に内蔵する DTU（データ転送コントローラ）を使用することができる．

■参考文献

1) F. Faggin, et al, “The history of the 4004,” IEEE Micro, vol.16, no.6, pp.10-20, Dec. 1996.
2) D. Alpert, “Architecture of the Pentium Microprocessor,” IEEE Micro, vol.13, no.3, pp.11-21, Jun. 1993.
3) S. Mirapuri, “The Mips R4000 Processor,” IEEE Micro, vol.12, no.2, pp.10-22, Apr. 1992.
4) M. C. Becker, “The PowerPC 601 Microprocessor,” IEEE Micro, vol.13, no.5, pp.54-68, Oct.1993.
5) S. P. Song, “The PowerPC 604 RISC Microprocessor,” IEEE Micro, vol.14, no.5, pp.8-17, Oct.1994.
6) G. Kurpanek, “PA7200: a PA-RISC processor with integrated high performance MP bus interface,” Compcon, pp.375-382, Feb. 1994.
7) E. McLellan, “The Alpha AXP Architecture and 21064 Processor,” IEEE Micro, vol.13, no.3, pp.37-47, Jun. 1993.
8) J. H. Edmondson, “Superscalar Instruction Execution in the 21164 Alpha Microprocessor,” IEEE Micro, vol.15, no.2, pp.33-43, Apr. 1995.
9) R. E. Kessler, “The Alpha 21264 Microprocessor,” IEEE Micro, vol.19, no.2, pp.24-36 Mar/Apr. 1995.
10) M. Muller, “ARM6: a high performance low power consumption macrocell,” Compcon, pp.22-26, Feb. 1993.
11) C. D. Snyder, “ARM Family Expands at EPF,” Microprocessor Report, Jun. 2002.
12) M. Baron, “ARM debuts logical v7,” Microprocessor Report, Nov. 2004.
13) M. Baron, “Cortex-A8: High Speed, Low Power” Microprocessor Report, Oct. 2005.
14) 河崎俊平, 他, “DSP 機能を内蔵した SH シリーズ,” 日経エレクトロニクス, no.568, pp.99-112, Nov. 1992.
15) S. Kawasaki, “SH-II A Low Power RISC Micro for Consumer Applications,” Hot Chips VI, pp.79-103, Aug. 1994.
16) A. Hasegawa, et al, “SH3: high code density, low power,” IEEE Micro, vol.15, no.6, pp.11-19, Dec. 1995.
17) O. Nishii, et al, “A 200MHz 1.2W 1.4GFLOPS microprocessor with graphic operation unit,” in Dig. Tech. Paper of the IEEE Int. Solid-State Circuits Conf. (ISSCC), pp.288-289, 1998.
18) 金子博昭, 他, “高性能･低消費電力動作の 32 ビット RISC シングルチップマイクロコンピュータ V851,” NEC 技報, vol.48, no.3, Mar. 1995.
19) K. Nodehira, et al, “Low-power multimedia RISC,” IEEE Micro, vol.15, no.6, pp.20-29, Dec. 1995.
20) M. R. Choudhury, et.al, “A 300 MHz CMOS microprocessor with multi-media technology,” in Dig. Tech. Paper of the IEEE Int. Solid-State Circuits Conf. (ISSCC), pp.170-171, 1997.
21) K. Diefendorff, “Pentium III = Pentium II + SSE,” Microprocessor Report, Mar. 1999.
22) J. Tyler, et al, “AltiVec™: bringing vector technology to the PowerPC™ processor family,” in Proc, IEEE Performance, Computing and Communications Conf. (IPCCC), pp.437-444, Feb. 1999.
23) F. Arakawa, et al, “SH4 RISC multimedia microprocessor,” IEEE Micro, vol.18, no.2, pp.26-34, Mar./Apr. 1998.
24) K. Krewell, “Intel Embraces Multithreading,” Microprocessor Report, Sep. 2001.
25) R. Kalla, et al, “IBM Power5 chip: A Dual-core Multithreaded Processor,” IEEE Micro, vol.24, no.2, pp.40-47, Mar./Apr. 2004.
26) D. Pham, et al, “The Design and Implementation of a First-Generation CELL Processor,” in Dig. Tech. Paper of the IEEE Int. Solid-State Circuits Conf. (ISSCC), pp.184-185, 2005.
27) A. S. Leon, et al, “A Power-Efficient High-Throughput 32-Thread SPARC Processor,” in Dig. Tech. Paper of the IEEE Int. Solid-State Circuits Conf. (ISSCC), pp.98-99, 2006.
28) S. Vangal, et al, “An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS,” in Dig. Tech. Paper of the IEEE Int. Solid-State Circuits Conf. (ISSCC), pp. 98-99, 2007.
29) S. Torii, “A 600MIPS 120mW 70μA Leakage Triple-CPU Mobile Application Processor Chip,” in Dig. Tech. Paper of the IEEE Int. Solid-State Circuits Conf. (ISSCC), pp. 88-89, 2008.
30) Y. Yoshida, et al, “A 4320 MIPS Four-Processor Core SMP/AMP with Individually Managed Clock Frequency

for Low Power Consumption,” in Dig. Tech. Paper of the IEEE Int. Solid-State Circuits Conf. (ISSCC), pp.100-101, 2007.

31) M. Nakajima, et al, “Low Power Techniques for Mobile Application SoCs Based on Integrated Platform "UniPhier",” in Proc. Asia and South Pacific design automation conference (ASP-DAC), pp.23-26, 2007.

■10 群 － 5 編 － 2 章

2-2 DSP

（執筆者：吉田　誠・久村孝寛）［2009 年 4 月 受領］

2-2-1 信号処理プロセッサの歴史

信号処理プロセッサ（DSP：Digital Signal Processor）はディジタル信号処理に適したプログラム可能なプロセッサである．ディジタル信号処理では積和演算が数多く使われるため，DSP は積和演算を高速に実行できるように構成されている．最初の DSP は通信用機器への応用を目的として 1980 年に登場した [1)-3)]．初期の DSP では，チップのメモリを除く回路の大半を乗算器が占めていた [1)]．当時のマイクロプロセッサはまだ乗算器を搭載していなかったため，1 クロックで乗算命令を実行可能な DSP はマイクロプロセッサと比較して 10 倍以上高速に乗算を実行することができた．更に，DSP は積和演算とデータ転送とを並列に処理できるように構成されており，まさしくディジタル信号処理に特化したプロセッサであった．

当初，DSP の主なアプリケーションは，音声符号化，データモデム，エコーキャンセラ，などであった．その後，通信速度の向上や無線通信技術の発展にともなって，無線通信のベースバンド処理や静止画や動画の符号化なども DSP の重要なアプリケーションとなっている（**図 2･5**）．近年では，多くの企業から低消費電力あるいは高い信号処理性能などをうたう DSP が提供されている [4)-7)]．DSP の端的な性能指標は積和演算を 1 秒間に何回実行できるかという数値（MMAC/sec：Million Multiply and Accumulate /sec）である．MMAC は DSP がもつ積和演算器の数と動作周波数で決定される．初期の DSP がもつ積和演算器（16 bit×16 bit の積を加算する演算器）の数は 1 個であったが，アプリケーションの要求性能が増加するにつれて，徐々に積和演算器の数が 2 個，4 個と増えてきた．更に多くの積和演算が必要なアプリケーションに対しては，動作周波数を向上させたり，複数の DSP を使用したり，専用回路を追加したりして，DSP の性能を向上させている．

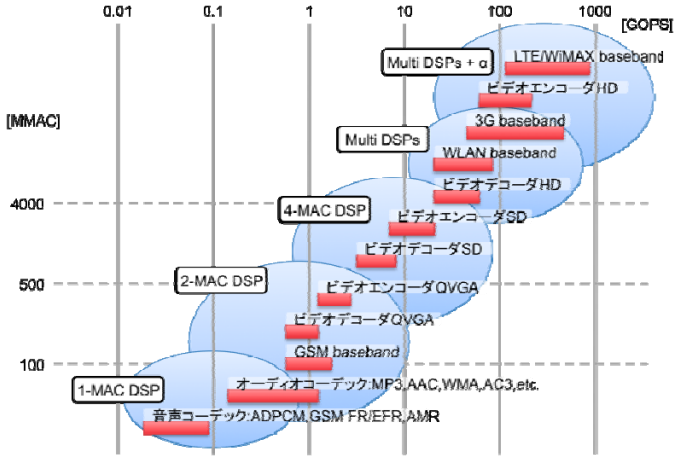


図 2･5　信号処理プロセッサのアーキテクチャとアプリケーション

2-2-2 信号処理プロセッサの構成

DSP の主な構成要素は，制御ユニット，データ演算ユニット，アドレス演算ユニット，命令メモリ，データメモリである（**図 2･6**）．DSP は命令メモリとデータメモリへ並列にアクセスできるように構成されている．命令メモリとデータメモリを明確に分離するメモリ構成はハーバードアーキテクチャと呼ばれている．DSP が登場したころのマイクロプロセッサでは命令メモリとデータメモリは分離されていなかったため，メモリ構成は DSP とマイクロプロセッサの違いのひとつであった．近年ではハーバードアーキテクチャがマイクロプロセッサでも一般的に使われている．

データ演算ユニットとアドレス演算ユニットは一般的なマイクロプロセッサとの違いが多いユニットである．データ演算ユニットはデータの演算を実行する演算ユニットであり，乗算器や加算器やデータ格納用レジスタを含む．データ格納用レジスタにはアキュムレータと呼ばれる乗算結果を連続的に加算するためのレジスタが含まれる．複数の乗算結果を連続的に加算してもオーバーフローしないように，アキュムレータのビット幅は乗算結果のビット幅よりも 8 ビット程度大きめに設定されている．

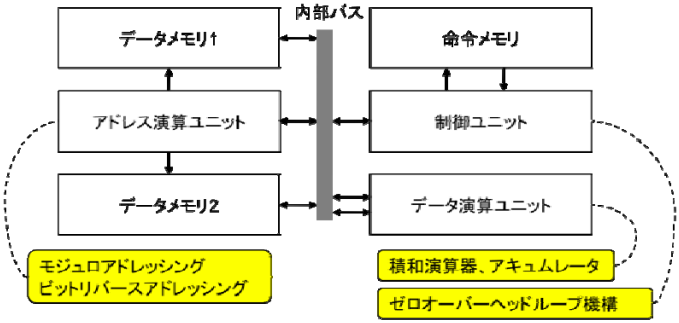


図 2・6　一般的な信号処理プロセッサの構成

データ演算ユニットは固定小数演算だけに対応したものと浮動小数演算にも対応したものとがある．現在 DSP において主流となっているのは固定小数演算方式である．固定小数演算方式の DSP ではソフトウェアで小数点の位置を管理する必要があるが，浮動小数演算方式に比べて固定小数演算方式の回路規模は小さいという利点がある．一方，浮動小数演算方式の DSP では，ハードウェアが個々の演算ごとに小数点位置を管理するため，プログラマが特別の考慮をしなくても演算精度を保ちながら広いダイナミックレンジを得ることができる．しかし，正規化処理を常に実行する浮動小数演算は固定小数演算に比べて演算結果を得るまでに必要なサイクル数（レイテンシ）が多い．こうした理由により，固定小数演算方式の DSP の方が広く使われている．固定小数演算方式の DSP においては，16 ビット固定小数の演算を主なターゲットとする DSP が一般的であるが，オーディオ処理には 24 ビット固定小数の演算器をもつ DSP が使われることもある．固定小数演算方式の DSP は，演算精度を保つために，丸めやスケーリングなどの機能を備えた積和演算器をもつのが一般的である．

アドレス演算ユニットはデータメモリのアドレスを計算するための演算ユニットであり，

加算器やアドレス格納用レジスタを含む．アドレス演算ユニットはディジタルフィルタや FFT（Fast Fourier Transform）などの信号処理アルゴリズムに登場するデータアクセスパターンを効率よく実現できるような演算器をもつ．そうした演算器によってモジュロアドレッシングやビットリバースアドレッシングといったアドレス更新が可能になる．モジュロアドレッシングは配列の循環的なアクセスに，ビットリバースアドレッシングは FFT におけるデータアクセスに，それぞれ適している．

DSP の制御ユニットは，ループ処理を効率よく処理するために，ゼロオーバヘッドループ機構をそなえる．ゼロオーバヘッドループ機構は，プログラムの中のある命令列を繰り返し実行するための命令実行機構である．条件分岐命令を使ってループ処理を行う場合には，プロセッサがループ処理の終端から先頭へ分岐する際にプロセッサが何も演算を実行しないサイクルが発生する．一方，ゼロオーバヘッドループ機構を使えば，プロセッサはループ処理の終端の命令を実行した直後にループ処理の先頭の命令を実行開始できる．ループ処理の 1 回のイタレーション当たりの命令数が少ない場合には，この機構はサイクル数の低減に特に効果的である．多くの DSP は多重のループ処理に対応したゼロオーバヘッドループ機構をもっている．

DSP の命令実行プロセスはいくつかのステージに分割されたパイプラインで構成されている．命令実行プロセスを複数のステージに分割して，各ステージを並列に動作させることによって，プロセッサの処理速度を向上させることができる．パイプラインは命令フェッチ，命令デコード，演算実行，メモリアクセス，演算結果書込みなどのステージで構成される．**図 2･7** に DSP とマクロプロセッサの実際のパイプラインの構成例を示す．図 2･7 の(a)から(c)の DSP [8)-10)] では，メモリアクセスステージの後に演算実行ステージが配置されている．これは多くの DSP に共通の特徴である．このようにパイプラインを構成することによって，メモリからレジスタへ転送されたデータが演算に使用可能になるまでにかかるサイクル数を低減できる．DSP はデータ用とアドレス用の分離されたレジスタファイルをもつために，このようなパイプライン構成が可能となる．一方，図(d) の組込み向けマイクロプロセッサ [11)] では，DSP とは異なり，演算実行ステージの後にメモリアクセスステージが配置されている．

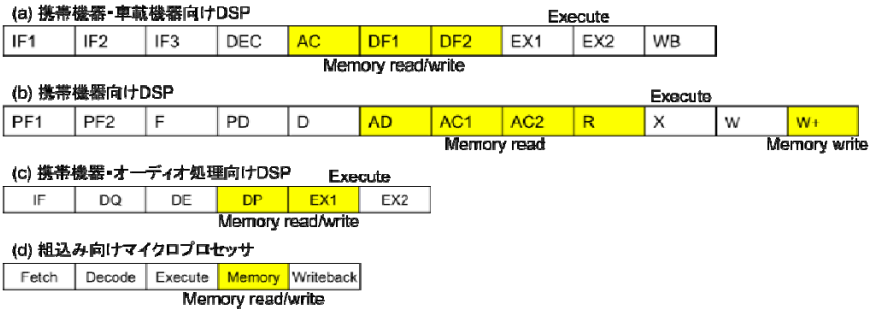


図 2･7　パイプラインの構成例

2-2-3 信号処理プロセッサの高性能化

近年の DSP は，汎用なプログラミング性能の向上，信号処理のさらなる高速化の二つの方向でアーキテクチャの研究開発が進んでいる．これらを実現するための一般的設計方針は，

(a) クロックの高速化，

(b) 1 クロック当たりの命令実行数の向上

の二つである．これらの設計方針のもとで各社の DSP は設計され，ターゲットとなるアプリケーションによって，演算器の数，レジスタの数，命令発行に関する制約，信号処理向けの複合命令体系などに DSP ごとの個性が現れる．ここでは，上記の二つの設計方針(a)と(b)をまず説明し，次に(b)の実現手段として多くの DSP が採用している VLIW 方式についてスーパースカラ方式と比較しながら説明する．

(1) クロックの高速化

クロックの高速化に関しては大きく分けてアーキテクチャ的なアプローチとより高速なプロセステクノロジーを選択するという二つの方法がある．アーキテクチャ的なアプローチとしてはディープパイプライン化や高速な回路構成への置き換えがある．ディープパイプライン化とは，パイプライン段数を増やすことによって，一つのパイプラインに必要な処理を分割し，ゲートの伝播遅延時間の合計を小さくし，高速なクロックを実現する手法である．ディープパイプライン化と高速な回路構成への置き換えはともに検討されることが多い．

パイプライン段数を多くするとクロックは高速化可能だが，分岐処理などのペナルティが大きく，総合的には性能が飽和してくる傾向がある．一方で，クロックの高速化にともなって消費電力は大きくなるので，消費電力の増大と性能向上のバランスが必要となる．消費電力の増大を抑制しながらクロックの高速化を行うために，プロセスの微細化とアーキテクチャの改善を同時に行うことが多い．

高速なプロセステクノロジーを選択する手法としては，より微細なプロセスを選択するか，同じ微細化世代の中でも，新材料・構造による高性能トランジスタや寄生容量低減を行った高速プロセステクノロジーを採用するといった方法がある．こういった高速プロセステクノロジーは一般にコストが高く，消費電力も大きくなることから，目的となる性能・コストに合わせて選択することになる．

(2) 1 クロック当たりの命令実行数の向上

1 クロックで実行可能な命令数を増加させる手法としては DSP では VLIW（Very Long Instruction Word）方式が多く採用され，少ないハードウェアオーバーヘッドでより高並列な命令発行を可能としている．一方，CPU ではバイナリ互換での性能向上を前提としてスーパースカラ（Super Scalar）方式が主流となっており，両者の思想の違いがアーキテクチャに現れているという意味で興味深い．**表 2･1** にスーパースカラ方式と VLIW 方式の特徴の比較を示す．1 クロックで実行可能な命令数を増やす手法に関しては，ほかに SIMD（Single Instruction Multiple Data stream）命令を実装する手法がある．SIMD 命令は複数のデータに対して同一の演算を実行する命令であり，データ並列度の高い信号処理には特に効果がある．

表 2・1　スーパースカラ方式と VLIW 方式の比較

	スーパースカラ 方式	VLIW 方式
並列度の検出	ハードウェア	ソフトウェア
互換性	バイナリ互換	ソース互換
想定並列度	2～4	4～8
命令デコーダや命令ディスパッチャの複雑度	並列実行可能な命令を抽出する回路が複雑	命令コードに埋め込まれた並列実行可能な命令の情報を使うため，回路は比較的単純

(3)　VLIW 方式とスーパースカラ方式

近年の高性能プロセッサは，VLIW 方式，スーパースカラ方式にかかわらず，一度に複数の命令をフェッチする．このようにフェッチした命令が**図 2・8** に示す入力命令ストリームとなる．入力命令ストリームから実際に実行する命令ストリームへの整列または並び替えが命令のディスパッチャまたは命令デコーダで行われる．このときに VLIW 方式ではどの命令が並列実行できるかが予め命令コード内に簡単にデコードできる形で埋め込まれており，ハードウェアが実行時に並列実行できるかどうかを抽出しない．このため，比較的容易に 8 並列程度の並列実行系を設計することが可能であり，並列度の高い信号処理系に向いたアーキテクチャを構築することが可能である．一般に VLIW 方式はスーパースカラ方式に比べてハードウェア実装が容易で高い並列度を作り込むことが可能である．

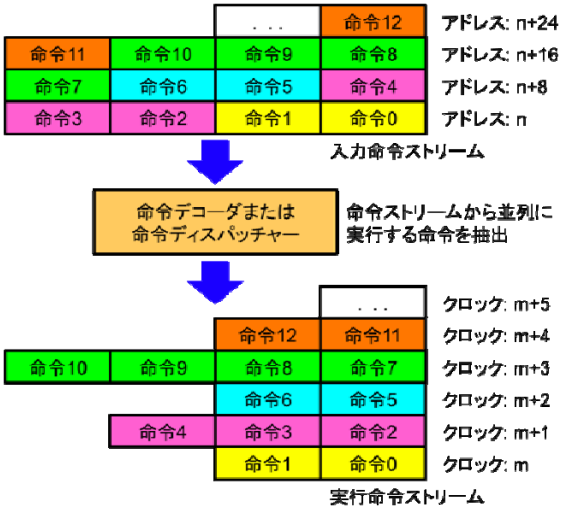


図 2・8　命令ストリームから並列実行可能な命令を抽出する

一方，スーパースカラ方式では，ハードウェアで命令の並列実行可能性を抽出するため，異なる並列度をもつハードウェアでは，プログラムの並列度が実行時に変わることになる．

並列実行可能な命令をどのように検出し，どのようにして並列に命令を実行するかによって，スーパースカラ方式のハードウェアには複雑度にかなり幅がある．後続命令が並列実行可能かどうかを判断するといった比較的実装が容易なレベルから，命令順序の入れ替えを行って並列実行可能な命令を抽出する高度なものがあり，回路規模や開発期間に大きな差がある．

性能チューニングをプログラム開発時点から行う場合は，どちらの方式でも，プログラムを書き換えたり，特定のプロセッサアーキテクチャに効率的なコード生成するためのオプションを切り替えて再コンパイルしたり，というアプローチは同様である．しかし，同一のオブジェクトコードの並列度が，実行するハードウェアの並列度に応じて実行時に変わるというスーパースカラ方式の柔軟性は，パッケージソフトウェアの流通という形態をもつ CPU には大きな利点であるといえる．一方，DSP は組込みプロセッサであり，消費電力，コスト当たりの性能の観点で優位な VLIW 方式が採用される傾向が強い．パッケージソフトウェアの流通性はあまり大きな問題でないため，ハードウェアの負担が大きいスーパースカラ方式はあまり採用されていない．

2-2-4 信号処理プロセッサとマイクロプロセッサの違い

近年では，多くの CPU が信号処理に適した命令を備えるようになり，多くの DSP はコンパイラに適したアーキテクチャに変わり，CPU と DSP は機能的に非常に近づいている．**表 2･2** に近年の CPU と DSP のアーキテクチャの概要を示す．表 2･2 によると，リアルタイム性の重要度や並列処理の実現手段などに違いがあることがわかる．DSP は実時間のディジタル信号処理を扱うので, DSP のハードウェアアーキテクチャや DSP で動作するミドルウェアやオペレーティングシステムを含むシステム全体がリアルタイム性を最優先課題として設計されている．これに対して，一般的には，CPU のハードウェアやソフトウェアはスループット最大化を最優先課題として設計されている．

表 2・2　近年の CPU と DSP のアーキテクチャの概要

	CPU	DSP
クロックの高速化	ディープパイプライン	ディープパイプライン
基本アーキテクチャ戦略	バイナリ互換での性能向上	回路規模を抑えて性能向上
互換性	バイナリ互換を維持	ソースレベル互換を維持
細粒度並列化方式	スーパースカラ方式（2〜4 way） SIMD 命令	VLIW 方式（4〜8 way） SIMD 命令
疎粒度並列化方式	対称型マルチプロセッサ	非対称型マルチプロセッサ
信号処理性能強化方法	乗算器の搭載，積和演算命令，飽和処理命令，SIMD 命令	2〜8 の乗算器の搭載 VLIW による高並列命令処理 SIMD 命令
プログラミング性能拡充のアプローチ	レジスタリネーミングによるレジスタ数の増加	多数のレジスタファイル コンパイラに適した命令
オペレーティングシステム	Windows，Linux などの仮想記憶 OS が主流	iTRON や Proprietary なリアルタイム OS が主流
システム性能最適化指標	スループット最大	リアルタイム性最優先

2-2-5 ソフトウェア開発ツール

DSP が登場した当初は，DSP の開発言語はアセンブリ言語だけであったが，やがて C コンパイラが開発され，現在では C 言語を標準的にほとんどの DSP で使用することができる．C++言語に対応したコンパイラが利用可能な DSP もいくつかある．ここでは，DSP のプログラムラム開発において重要なツールであるアセンブラと C/C++コンパイラについて説明する．

(1) アセンブラ

DSP のプログラムはアセンブラで開発されることが多い．そのために，一般的なプロセッサのアセンブラに比べて，DSP のアセンブラは高度な機能をもつ．多くの DSP のアセンブラの命令シンタックスは算術記述形式である．算術記述形式とは，代入記号 = の左辺に命令のデスティネーションオペランドを，右辺に命令の動作を表す式をソースオペランドとともに記述する記述方法である．ニモニックとオペランドを記述する方法に比べて，算術記述形式は命令の動作や演算結果の格納先を理解しやすい．算術記述形式のプログラムの例を**図 2･9** に示す．図 2･9 は三つの異なる DSP のアセンブラ言語で記述した FIR フィルタのプログラム例である [8)-10)].

(a) 携帯機器･車載機器向けDSPのプログラム例

```
A0=0 || R0.L = W[I0++]  || R1.L = W[I2--];
LSETUP(LOOP1B,LOOP1E) LC1 = P1;
LOOP1B:
  R7.L=(A0+=R0.L*R1.L) || R0.L = W[I0++]  || R1.L = W[I2--];
  R7.L=(A0+=R0.L*R1.L) || R0.L = W[I0++]  || R1.L = W[I2--];
LOOP1E:
```

(b) 携帯機器向けDSPのプログラム例

```
AC0 = #0 ; Clear AC0
repeat(CSR) ; Repeat the following lines
  AC0 = AC0 + ( *AR0- * coef(*CDP+) )
```

(c) 携帯機器･オーディオ処理向けDSPのプログラム例

```
loop R7L;
nop;
nop;
R1L=*dp1++, R2L=*dp2--;
{
  R0=R0+R1L*R2L, R1L=*dp1++, R2L=*dp2--;
  R0=R0+R1L*R2L, R1L=*dp1++, R2L=*dp2--;
}
```

図 2･9　算術記述形式のアセンブリ言語の例

アセンブラ言語でプログラムを開発する際には，プログラマは，複数の処理の組合せだけでなく，パイプラインハザードにも注意しなければならない．DSP の回路を大きく複雑にしないために，ソフトウェア的にパイプラインハザードを回避する方法が選択されることが多い．パイプラインハザードはパイプラインの連続的動作を中断させたり，パイプラインを正常に動作できなくしたりするので，プログラムを最適化する際にプログラマはパイプラインハザードが発生しないように注意を払う必要がある．

パイプラインハザードをソフトウェア的に回避するために，NOP 命令を必要な場所に自動的に挿入したり，プログラムの意味を変えることなく命令スケジューリングを行ったりするアセンブラもある．更には，コンパイラのように，レジスタ割当て機能をもつアセンブラもある [12]．レジスタ割当て機能をもつアセンブラの言語仕様においては，プログラマは実際のレジスタの代わりにレジスタ変数を使うことができる．このタイプのアセンブラはレジスタ変数に適切なレジスタを割り当てるので，プログラマはレジスタ割当ての煩わしさから解放される．更に，このタイプのアセンブラはプログラムの各命令が使用する演算ユニットやレジスタなどのリソースを分析し，各命令の依存関係を把握し，命令並列化や命令スケジューリングなどを行う．つまり，高機能なアセンブラはコンパイラのバックエンド部とほぼ同じ機能をもつ．高機能アセンブラを使うことによって，C/C++言語では表現できない DSP 独特の演算を効率よく表現することができる．

(2) C/C++コンパイラ

初期の DSP のプログラムはすべてアセンブリ言語で記述されていたが，今では制御処理が多い部分を C 言語で記述し，信号処理部分をアセンブリ言語で記述する，という開発スタイルが一般的になっている．このような開発スタイルになった背景には，(a) ソフトウェアの規模が大きくなりアセンブリ言語だけで開発することが困難になったこと，(b) DSP のアーキテクチャが一般的なプロセッサに近づきコンパイラを開発しやすくなったこと，(c) コンパイラの最適化が進んで実用に堪えるコードをコンパイラが生成できるようになったこと，などがある．

DSP に特有のコンパイラ最適化には，ポインタ更新処理最適化，ループ処理最適化，データ配置最適化などがある．ポインタ更新処理最適化とは DSP がもつアドレッシングモードを活用するための最適化である．ループ処理最適化において，コンパイラはプログラムに含まれるループ処理に可能な限りゼロオーバヘッドループ機構を適用することを試みる．データ配置最適化は，DSP の並列アクセス可能な複数のデータメモリ領域を活用するための最適化である．こうした最適化を駆使してコンパイラは DSP のコードを生成する．

ディジタル信号処理では固定小数演算や飽和処理などが使用されるが，C/C++の言語仕様はディジタル信号処理に特有のこれらの処理の表現には適していない．そこで，多くの DSP のコンパイラは固定小数演算や飽和処理を表現するための独自のフレームワークをもっている．例えば，固定小数演算のための C 言語の慣用句的記述が規定されていたり，DSP の命令に一対一に対応した組込み関数で固定小数演算を表現したりする．一方で，こうしたフレームワークとは別に，信号処理に適した C/C++の言語仕様 Embedded C を策定するという活動も行われている [13]．Embedded C は固定小数型やアキュムレータ型や飽和演算などの仕様が追加された C 言語である．Embedded C の利点は，高級言語を使用して固定小数演算を表現できることと，DSP のコードの移植性を高められることである．

2-2-6 プログラム開発手順

DSP のプログラムを新規に開発する場合には，プログラムとして記述される信号処理アルゴリズムの動作を検証するためのリファレンスプログラムを C/C++言語あるいは MATLAB などを使ってまず作成する（**図 2･10**）．このアルゴリズム検討用リファレンスプログラムを

一般の PC で動作させ，プログラマはどのようなアルゴリズムを使えばよいかを検討する．

続いて，アルゴリズム検討用リファレンスプログラムを DSP へ実装する前に，DSP の演算性能や演算精度などを考慮して，実装用リファレンスプログラムを作成する．実装用リファレンスプログラムは DSP で最終的に動作するプログラムと全く同じ演算結果を出力するプログラムである．実装用リファレンスプログラムは一般の PC，及び DSP の両方で動作することが望ましいので，C 言語で記述されることが多い．

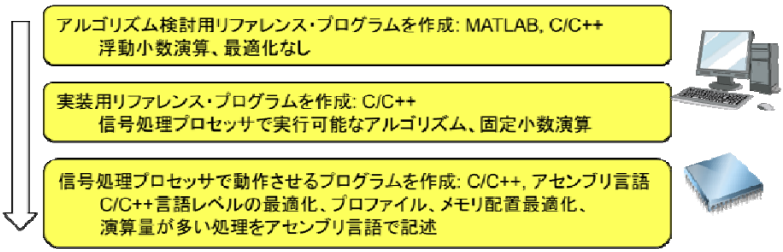


図 2・10　信号処理プロセッサのためのプログラム開発手順

そして，この実装用リファレンスプログラムに基づいて，DSP で動作させるための最終的なプログラムを作成する．このような手順でプログラムを開発することによって，プログラムのデバッグや動作確認が容易になる．もしアルゴリズム検討と DSP 特有の最適化を同時に行うと，不具合の原因を特定することが困難になる．プログラム開発の効率に関しては，実際の LSI よりもシミュレータの方が，DSP よりも一般的な PC の方がデバッグは容易である．したがって，開発が容易な環境でできるだけ作業を進め，なおかつ，困難なデバッグを避けるために二つのリファレンスプログラムを作成するのである．

DSP の上で実行されるプログラムの数が増えたり構成が複雑になったりするにつれて，DSP でもオペレーティングシステム（OS : Operating System）を使うことが増えている．DSP で使用される OS は割込み管理機能や排他制御機能やタスク制御機能などを備えたリアルタイム OS である．代表的なものは，DSP とともに提供される DSP 専用 OS や，μITRON 互換の OS やμCLinux である．μCLinux はもともと仮想記憶機構をもたないマイクロプロセッサ向けに作られた，仮想記憶なしで動作する Linux である．DSP が一般のマイクロプロセッサのアーキテクチャに近づき，DSP のコンパイラが整備されたために，DSP で Linux を動作させることが可能になっている．リアルタイム性を重視する場合には DSP 専用 OS やμITRON が，マイクロプロセッサ上のソフトウェアを移植したい場合などにはμCLinux が適している．

2-2-7 SoC の中の DSP サブシステム

近年の半導体製造技術の向上や設計ツールの高機能化によって，従来は複数のチップで構成されていたシステムが一つのチップに集積されるようになった．このようなチップを SoC（System-on-a-Chip）と呼ぶ．複数の構成要素を一つのチップに集積することによって，専有面積の減少，動作周波数の高速化，低消費電力化，部品点数削減，などの様々なメリットが得られる．この集積化の流れの中で，ほかの構成要素と同じく，DSP も一つの機能ブロックとして SoC の中に組み込まれるようになった．SoC の設計では，各機能ブロックの再利用性

が重要視され，各機能ブロックは理解しやすいインタフェースを備えている．複数の機能ブロックが互いに依存関係をもつような複雑な設計は好まれず，複雑な機能をもつ回路は一つのブロックとして設計されることが多い．

DSP をその中に含む SoC においては，一般的には，DSP はメモリと外部へのインタフェースと DSP とで構成される DSP サブシステムとして存在している（**図 2･11**）．この DSP サブシステムは再利用可能な形で設計されている．DSP サブシステムは AXI や OCP といったオンチップバスの標準インタフェースを備えている．そのようなインタフェースを使って DSP は SoC のほかの機能ブロックにアクセスすることができる．

DSP サブシステムに含まれるメモリがキャッシュであることもある．従来，キャッシュはリアルタイム性を損なうものとして DSP ではあまり用いられてこなかった．しかし，DSP が扱うデータのサイズや DSP のプログラムサイズが大きい場合には，そのすべてを DSP サブシステムのメモリに格納することが困難なので，キャッシュが使われる．キャッシュを使う場合には，実用的なリアルタイム性を損なわないように，キャッシュのサイズやバス調停機構などが慎重に設計される．

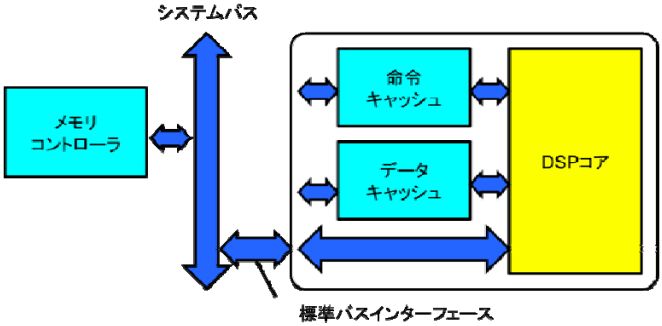


図 2・11　SoC の中の DSP サブシステム

■参考文献

1) Y. Kawakami et al, “A Single-Chip Digital Signal Processor for Voiceband Applications,” IEEE International Solid-State Circuits Conference 1980, Digest of Technical Papers, vol.XXIII, pp.40-41, Feb. 1980.
2) 西谷, “DSP の誕生とその発展(前編),” 電子情報通信学会 基礎・境界ソサイエティ Fundamentals Review, vol.1, no.4, Apr. 2008, available at http://w2.gakkai-web.net/gakkai/ieice/vol4pdf/vol1no4_17.pdf
3) 西谷, “DSP の誕生とその発展(後編),” 電子情報通信学会 基礎・境界ソサイエティ Fundamentals Review, vol.2, no.1, Jul. 2008, available at hhttp://w2.gakkai-web.net/gakkai/ieice/vol5pdf/vol2no1_9.pdf
4) Robert Cravotta, “The 2008 EDN DSP directory,” EDN magazine, Mar. 20, 2008, available at http://www.edn.com/toc-archive/2008/20080320.html
5) Berkeley Design Technology Inc., “Choosing a DSP Processor,” BDTI White Paper, 2000, available at http://www.bdti.com/articles/choose_2000.pdf
6) Berkeley Design Technology Inc., “Evaluating DSP Processor Performance,” BDTI White Paper, 2002, available at http://www.bdti.com/articles/benchmk_2000.pdf
7) Berkeley Design Technology Inc., “DSPs Adapt to New Challenges,” BDTI White Paper, 2003, available at http://www.bdti.com/articles/DSPs_Adapt.pdf
8) Analog Devices Inc., Blackfin Processor Programming Reference, Rev.1.3, Sep. 2008.

9) Texas Instruments Inc., TMS320C55x DSP CPU Reference Guide, SPRU371F, Feb. 2004.
10) T. Kumura et al., “VLIW DSP for mobile applications,” IEEE Signal Processing Magazine, vol.19, Issue 4, Jul. 2002.
11) ARM Ltd., ARM9E-S Technical Reference Manual, Rev.2, May 2002.
12) Texas Instruments, “TMS320C6000 Optimizing Compiler v6.1 User’s Guide,” SPRU1870, May 2008, available at http://focus.tij.co.jp/jp/lit/ug/spru187o/spru187o.pdf
13) ISO/IEC JTC1 SC22 WG14 N1169, “Programming languages - C – Extensions to support embedded processors,” ISO/IEC TR 18037, Apr. 2006, available at http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1169.pdf

■10 群 － 5 編 － 2 章

2-3 アクセラレータ

（執筆者：越智裕之）［2009 年 11 月 受領］

画像や音声の符号化に代表される信号処理や，無線ベースバンド処理に代表される通信処理では，実時間制約下で多量のデータを処理することが求められる．更に，携帯情報機器などへの応用では消費電力に対する制約も課される．性能や消費電力を重視すれば，処理内容に特化した専用ハードウェアエンジンとして実現することが有力な選択肢となるが，開発工数，及びマスクコストがかかる欠点がある．また，アプリケーションや処理内容別に専用ハードウェアエンジンを実装すれば，全体の回路規模が大きくなってしまう．こういったコストの観点では，プログラマビリティをもったアーキテクチャが望ましい．

本節で紹介するリコンフィギャラブルプロセッサや SIMD アレイプロセッサは，ソフトウェア（構成情報）の開発のみによりアプリケーションが実装でき，これを実行時に切り替えることによって限られたハードウェア資源で様々な処理内容に対応できる一方，並列動作可能な多数の演算器とそれに見合った配線資源やメモリを実装することによって必要な性能を確保し，アプリケーションドメインをある程度絞ることによって回路面積や消費電力の無駄を抑えている．

2-3-1 リコンフィギャラブルプロセッサ

リコンフィギャラブルデバイス〔6 群 5 編 6 章参照〕は，機能の書き換えにより様々な回路を実現できる集積回路である．様々なアーキテクチャが提案されているが，多数のプログラマブルな演算要素セルがアレイ状に配置され，その間にプログラマブルな配線資源が用意されているのが一般的である．演算要素セルが論理演算レベルでプログラム可能な LUT などで実現されているものが FPGA であるが，高い自由度と引き換えに，面積，性能，消費電力で多大な代償を払っている．例えば，純粋な LUT ベースの FPGA に同一世代のプロセスの ASIC と同じ回路を実装した場合，平均で面積が 35 倍，遅延時間が 3.4～4.6 倍，消費電力が 14 倍になったという報告がある [1]．

このギャップを埋めるため，演算要素セルのプログラムを算術演算のレベルとし，配線資源のプログラムも決められた語長（4 ビットから 32 ビットまで様々なアーキテクチャがみられる）を単位として行うようにしたのが粗粒度リコンフィギャラブルデバイスである．演算器や配線資源の粒度を高めることで，最適化された回路や物理構造を作り込んでおくことが可能となる上，構成情報を保持するためのメモリ容量も大幅に小さくすることができる．構成情報量を小さく抑えることは，後述する動的再構成を実現する上でも重要である．

あらゆる応用に適合する粗粒度リコンフィギャラブルアーキテクチャはあり得ない [2]．演算要素セルがサポートする演算の種類や語長などにより，適合するアプリケーションドメインはある程度限定される．例えば FFT を並列実装する場合，乗算をサポートする演算要素セルと，バタフライ演算を実装するためのリッチな配線資源が不可欠である．逆に，乗算をほとんど使わないアプリケーションを実装する場合，多数の演算要素セルに乗算器を内蔵したアーキテクチャは面積の観点で不利である．対象とするアプリケーションドメインを絞り，必要な要素は盛り込みながら，面積，性能，消費電力の無駄をなくすことが肝要である．

以下，リコンフィギャラブルプロセッサの例としてアイピーフレックスの DAPDNA-2 [3]，NEC エレクトロニクスの DRP-1 [4]，並びに日立の FE-GA [5] の概要を**表 2･3** に示す．

表 2･3　リコンフィギャラブルプロセッサの例

製品名	基本語長	アレイ構造	乗算器	接続網
DAPDNA-2	32 ビット	ヘテロジニアス	演算 PE 168 個中 56 個	アイランドスタイル
DRP-1	8 ビット	ホモジニアス	アレイ外に 8 個	アイランドスタイル
FE-GA	16 ビット	ヘテロジニアス	演算 PE 32 個中 8 個	直結型・クロスバ併用

基本語長は表 2･3 の例を見ても 8 ビットから 32 ビットまで様々である．ただし，DRP-1 は隣接セルを組み合わせて多バイト演算を行うことができる．アレイ構造は，表 2･3 の例では DRP-1 のみホモジニアスであるが，最近の例では東芝の FlexSword も 8×5 のホモジニアスな PE アレイをもっている [6]. 演算 PE に占める乗算 PE の比率は, DAPDNA-2 が 1/3, FE-GA が 1/4 である．接続網は，PE アレイを縦断，横断する配線資源に PE が接続する形のものを FPGA にならってアイランドスタイルに分類した．FE-GA は，PE アレイ内では隣接 PE とのみ直結されているが，アレイ外周に柔軟なクロスバネットワークをもっている．再構成の方法は，構成情報をその都度アレイ外部から流し込む構成情報配送方式と，予め各 PE に複数の構成情報を格納しておき，コントローラからの指示で瞬時に切り替えるマルチコンテキスト方式があるが，表 2･3 にあげた三つのリコンフィギャラブルプロセッサはいずれも後者である．

2-3-2 SIMD エンジン

画像処理では，取り扱う画素数の増大にともない，特に高い並列処理性能が必要となってきた．画像認識の前処理で必要となるフィルタ処理などでは，多数の画素に同一の演算を施す SIMD 演算が処理の大半を占めるため，これを高速に実行するべく，SIMD 演算命令を備えたプロセッサが多数開発されてきた. しかし, 更に性能や電力効率を向上させるためには, SIMD 処理に特化したハードウェアエンジンが有力である．

ここでは非常に多くの PE を集積し，これらに同じ命令を供給して高並列な処理を実行できるアーキテクチャを SIMD エンジンと呼ぶことにする．SIMD エンジンの大きな特徴は，非常に大容量の SRAM を同一チップ上に実装して PE に直接データを供給する構造になっているということである．

SIMD エンジンの例として，NEC の IMAP-2 [7] とルネサステクノロジの MTX [8] の概要を**表 2･4** に示す．

表 2･4　SIMD エンジンの例

製品名	PE	PE 数	メモリ構成	ピーク性能
IMAP-2	8 ビットプロセッサ	64 個	8 bit×4K word×64 PE	3.84 GIPS （8 bit 演算@40 MHz）
MTX	2 ビット演算器	2 048 個	128 bit×4 column×2048 PE	40 GOPS （16 bit 加算@200 MHz）

IMAP-2 は 8 ビットデータを処理するプロセッサを 64 個有する．各プロセッサは 12 語の汎用レジスタのほか，インデックスレジスタなどをもつ．この 64 個のプロセッサが並列にそれぞれ 4 k 語のメモリをアクセスしながら処理を行う．0.55 μm BiCMOS プロセスで製造された IMAP-2 はその後改良され，IMAPCAR では 100 GOPS を達成している〔本編 3 章 3-2 参照〕．

MTX は 2 ビット演算器でビットシリアル的にデータを処理する．PE がコンパクトで多数の PE を 1 チップ上に実装可能であり，また任意語長のデータを扱える利点もある．

2-3-3 まとめ

本節ではプログラマビリティをもち，多数の演算器を並列に動作させることで信号処理や通信処理に必要な高い性能を達成するアーキテクチャとして，リコンフィギャラブルプロセッサと SIMD エンジンを取り上げた．いずれもプログラマビリティはもっているが，アプリケーションドメインに特化して PE 機能や PE 数などがチューニングされ，それぞれに特徴のあるアーキテクチャとなっている．また，高いスループットを達成するためには，PE アレイの入出力バンド幅がそれに見合ったものになっている必要があり，配線資源やメモリ資源の構成も工夫されている．ただし，一つのアプリケーションの中にも性質の異なる様々な処理が必要であり，必ずしも単一のアーキテクチャですべてを効率よく実現できるとは限らず，目的に応じて複数のアーキテクチャを組み合わせることが必要になるであろう．幸いにもワンチップの集積回路に実装可能な回路規模はますます増大しているため，そのようなヘテロジニアスマルチコアアーキテクチャは現実的な選択肢となりつつある．

■参考文献

1) I. Kuon and J. Rose, “Measuring the Gap Between FPGAs and ASICs,” IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol.26, no.2, pp.203-215, 2007.
2) R. Hartenstein, “Coarse Grain Reconfigurable Architectures,” Proc. ASP-DAC 2001, pp.564-570, 2001.
3) T. Sugawara, K. Ide, and T. Sato: “Dynamically Reconfigurable Processor Implemented with IPFlex’s DAPDNA Technology,” IEICE Trans. Inf. & Syst., vol.E87-D, no.8, pp.1997-2003, 2004.
4) M. Motomura, “A Dynamically Reconfigurable Processor Architecture,” Proc. Microprocessor Forum, 2002.
5) T. Kodama, T. Tsunoda, M. Takada, H. Tanaka, Y. Akita, M. Sato, and M. Ito, “Flexible Engine: A Dynamic Reconfigurable Accelerator with High Performance and Low Power Consumption,” Proc. COOL Chips IX, pp.393-408, 2006.
6) T. Yoshikawa, “A Dynamically Reconfigurable Architecture for Stream Processing,” ICFPT 2007 Tutorial, 2007.
7) 藤田善弘, 山下信行, 木村　亨, 中村和之, 岡崎信一郎, “メモリ集積型 SIMD プロセッサ IMAP”, 信学論(D-I), vol.J78-D-I, no.2, pp.82-90, 1995.
8) Noda, H.; Nakajima, M.; Dosaka, K.; Nakata, K.; Higashida, M.; Yamamoto, O.; Mizumoto, K.; Tanizaki, T.; Gyohten, T.; Okuno, Y.; Kondo, H.; Shimazu, Y.; Arimoto, K.; Saito, K.; Shimizu, T.: “The Design and Implementation of the Massively Parallel Processor Based on the Matrix Architecture,” IEEE J. Solid State Circuits, vol.42, no. 1, pp.183-192, 2007.

■10 群 － 5 編 － 2 章

2-4 ASIP

（執筆者：水野　淳）［2008 年 12 月 受領］

2-4-1 ASIP とコンフィギュラブルプロセッサ

ASIP（Application Specific Instruction Set Processor）とは，命令セットやメモリシステムなどのアーキテクチャが特定の応用分野に適するように設計されたプロセッサのことであり，特定応用分野向けプロセッサと訳されることもある．汎用プロセッサのもつ柔軟性と，ASIC の高性能を兼ね備えた存在といえる [1]．

しかし，応用分野ごとに ASIP をゼロから設計していては開発効率がよくない．このため，命令の追加・削除や，キャッシュメモリのサイズの選択など，機能や構成を設計時に変更できるプロセッサが考案された．このようなプロセッサをコンフィギュラブルプロセッサという．コンフィギュラブルプロセッサを用いた LSI 開発では，設計時に機能や構成を変更することで，製品に適したプロセッサ（＝ASIP）を使用することができる．

コンフィギュラブルプロセッサにおいて設計時に変更可能な機能や構成のことをコンフィギュレーションと呼ぶことにする．コンフィギュラブルプロセッサでよく見受けられるコンフィギュレーションを**表 2･5** に示す．

表 2・5　コンフィギュラブルプロセッサのコンフィギュレーションの種類の例

主なコンフィギュレーション	具体例
命令の選択	乗算命令の有無 マルチメディア専用命令の有無
メモリシステム	キャッシュメモリのサイズ，ウェイ数，ライトバック方式の選択
インタフェース	データのビット長の選択 バスプロトコルの選択
機能拡張	既存のコプロセッサの付加 ユーザ定義命令の追加 ユーザ定義のハードウェアモジュールの付加（ハードウェア拡張）

コンフィギュラブルプロセッサの特色として開発環境もコンフィギュラブルであることがあげられる．開発環境については次の節で説明する．

コンフィギュラブルプロセッサと混同しそうな用語としてリコンフィギュラブルプロセッサという用語があるが，両者は異なる概念である．コンフィギュラブルプロセッサが設計時に構成の変更を行うことに対し，リコンフィギュラブルプロセッサは LSI になって回路が動作している間に構成を変えることができる．つまりコンフィギュラブルプロセッサは LSI になってしまえば構成を変えることはできない．コンフィギュレーションが確定した後のコンフィギュラブルプロセッサは，LSI の開発フローや LSI の動作原理などは従来の汎用プロセッサと同じである．

2-4-2 コンフィギュラブルプロセッサの例

本節では代表的なコンフィギュラブルプロセッサとして，東芝の MeP と Tensilica の Xtensa を取り上げる．

(1) MeP

MeP（Media embedded Processor）は東芝が開発したマルチメディア処理用のコンフィギュラブルプロセッサである [2]．オプション命令の選択，キャッシュメモリのサイズの選択などが可能であり，またユーザが独自に定義した命令の追加や，特定の処理を行うハードウェア（ハードウェアエンジン）を付加するハードウェア拡張機能も備えている．コアプロセッサとハードウェアエンジンなどの機能拡張部を組み合わせたモジュールを MeP モジュールと呼び，アプリケーションに特化したプロセッサモジュールの単位としている（**図 2･12**）．

MeP は多くのマルチメディア系 SoC に採用されている．**図 2･13** は MeP を搭載した画像処理 LSI のチップ写真である．

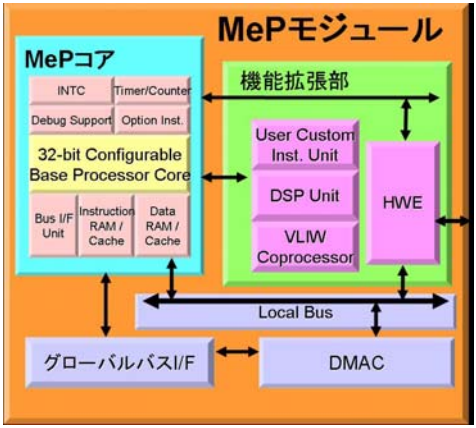


図 2･12　MeP モジュールの構成

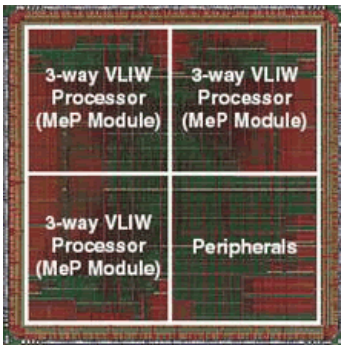


図 2･13　MeP を搭載した画像処理 LSI Visconti ™

(2) Xtensa

Xtensa は Tensilica 社が開発したコンフィギュラブルプロセッサである [3]．乗算命令などのコンフィグレーション項目が選択できるほか，命令やレジスタの追加などの機能拡張が可能である．命令の追加には，TIE（Tensilica Instruction Extension）と呼ばれる独自の言語を用いて命令の仕様を定義する．これらのコンフィグレーションの指定や TIE 言語による命令定義を Xtensa Processor Generator と呼ぶ GUI のツールに与えることで，所望の機能をもつ Xtensa プロセッサの RTL 記述を生成させることができる．また，同時にコンパイラなどのソフトウェア開発ツールもコンフィグレーションに応じたものが生成される．

紹介した二つのプロセッサ以外にも，ARC 社などがコンフィギュラブルプロセッサを発表している．

2-4-3 コンフィギュラブルプロセッサの問題点と今後

これまでコンフィギュラブルプロセッサの特長を述べてきたが，ここではコンフィギュラブルプロセッサの問題点について説明し，それを踏まえて今後のコンフィギュラブルプロセッサの方向性について考察する．

コンフィギュラブルプロセッサの問題点の一つはソフトウェアの再利用の問題である．既存の LSI 用に開発されたソフトウェアを別の LSI に流用する場合を考える．同じプロセッサを用いた LSI 同士であれば，既存のソフトウェアをそのまま流用できる（ここでプロセッサ以外のハードウェアに依存したソフトウェア処理の部分は除外して考える）．しかし，コンフィギュラブルプロセッサの場合，コンフィギュレーションの設定が異なれば同じプロセッサを使用した LSI 同士でも，一方で動作したソフトウェアは他方では動作しない可能性がある．例えば，それぞれのコンフィギュレーションで異なる追加命令を使用している場合，バイナリプログラムをそのまま流用することはできない．少なくともソースプログラムを再コンパイルする必要があるが，追加命令をコンパイラが生成しやすいようにソースプログラムを書き換える作業が発生する場合が多い．LSI の開発期間の中でソフトウェア開発の占める期間が増大していることを考えると，プログラムを書き換える作業はできればない方が望ましい．コンパイラ技術の発展が重要であり，このことは後ほど述べる．

別の問題点として，コンフィギュラブルプロセッサの検証の問題がある．変更可能な機能とその選択可能な値が増えるにつれ，コンフィギュレーションの組合せは爆発的に増加し，すべての組合せで検証することは不可能となる．これはプロセッサのハードウェアの検証だけでなく，ソフトウェア開発ツールの検査にも当てはまる．

これらのことから，コンフィギュラブルプロセッサの今後の傾向として，変更可能な機能はあまり増えず，逆に実際によく使用されるコンフィギュレーションは限定されていくと思われる．また近年，汎用のプロセッサにおいても，キャッシュメモリのサイズなどはいくつかの値を選択できるようなプロセッサが現れてきている．したがって今後は，コンフィギュラブルプロセッサと汎用プロセッサの境界は明確でなくなり，汎用プロセッサが ASIP 的な要素を備える方向にいくと予測される．

2-4-4 ASIP の開発環境

(1) 組込みシステムの開発環境

コンフィギュラブルプロセッサの開発環境は，後で述べるように通常のプロセッサの開発環境とは異なる特長がある．それを説明するために，まず最初に組込みプロセッサの開発環境一般の話を説明する．

組込みシステムのソフトウェア開発と PC 用のソフトウェア開発と異なる点は，開発環境と最終製品のソフトウェアが実際に動作する環境が異なる点である [4]．PC 用のソフトウェア開発が，PC にインストールされているエディタやコンパイラを使用してソフトウェアを開発し，PC 上でそのソフトウェアが動作することを確認すればよい．しかし組込み機器では，プログラムを作成するのに適したキーボードやディスプレイは装備されておらず，コンパイラも搭載されていない．このため組込みソフトウェア開発は，最終製品とは別の開発環境（通常は PC）でソフトウェアを開発し，出来上がったソフトウェア（組込みプロセッサ用のバイナリコード）を実機にダウンロードし，最終確認を行うことになる．開発環境として使う PC をホストマシン，その PC のプロセッサをホストプロセッサと呼び，製品に搭載されるプロセッサをターゲットプロセッサと呼ぶ．ターゲットプロセッサ用のコンパイラはホストマシン上で動作しており，このように自身が動作しているプロセッサと異なるプロセッサのバイナリコードを生成するコンパイラをクロスコンパイラと呼ぶ．

LSI の開発期間の短縮化のためには，実機が完成する前にソフトウェアを開発する必要がある．このために，ホストマシン上でターゲットプロセッサ用のバイナリプログラムを解釈して実行するソフトウェアを用いて，ホストマシン上でデバッグや動作確認を行うことが一般的になっている．このようなソフトウェアを命令セットシミュレータ（Instruction Set Simulator : ISS）と呼ぶ．また，ISS 上で動作しているプログラムを停止させたり状態を変更させたりしてデバッグを行うためのソフトウェアをソフトウェアデバッガと呼ぶ．なお，ISS を以降では単にシミュレータと呼ぶ．また，プロセッサモデルという用語も ISS と同義で用いられる．

(2) コンフィギュラブルプロセッサの開発環境の特長

コンフィギュラブルプロセッサのソフトウェア開発環境の特長として，コンパイラ，アセンブラ，シミュレータ，デバッガなどのソフトウェア開発ツールが，LSI 設計時に指定されたコンフィグレーションの設定によりそれに応じて機能が変わるということである．したがって，コンフィギュレーションの設定によりソフトウェア開発ツールをカスタマイズするツールが存在する（**図 2･14**）．Xtensa の開発環境の Xtensa Processor Generator がこの機能に相当する．

なお，命令追加機能に関連して，リターゲッタブルコンパイラなどの先端コンパイラ技術が必要となり，またハードウェア拡張機能と関連してハードウェア・ソフトウェア協調シミュレーション技術と関係してくる．これらについて以降で詳しく説明する．

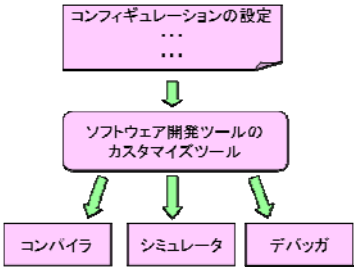


図 2・14　コンフィギュラブルプロセッサの開発環境

(3)　コンパイラ技術

コンフィギュラブルプロセッサの問題点の説明でも述べたように，ソフトウェアの再利用問題を解決するにはコンパイラ技術が非常に重要である．乗算命令の有無のように，予め決められた命令の選択であれば，コンパイラにも予めそれらの命令生成の機能を入れておき，コンパイラ実行時にコンフィギュレーションの設定からそれらの命令の生成有無を判断させるようにすればよく，コンパイラの実装は比較的容易である．

しかし，ユーザ定義命令の場合，リターゲッタブルコンパイラの技術が必要となる．後述する ASIP Meister という ASIP 開発環境は，プロセッサの仕様を入力してコンパイラを生成する機能を有している．

また，並列化コンパイラ技術も必要である．コンフィギュラブルプロセッサのコンフィギュレーションとしてコプロセッサを付加する場合がある．このとき，コプロセッサが SIMD 型である場合や，プロセッサ本体と合体して VLIW 型の命令となる場合もある．このようなコプロセッサが付加された場合，ソースプログラムを極力変更せずに，SIMD や VLIW の命令を利用したバイナリプログラムを生成するコンパイラが望まれており，研究がなされている．

(4)　ハードウェア・ソフトウェア協調シミュレーション

コンフィギュラブルプロセッサにハードウェア拡張機能がある場合，シミュレータにハードウェア・ソフトウェア協調シミュレーション機能が必要である．しかし，ほとんどの LSI がハードウェアとソフトウェアが混在して構成される現在，コンフィギュラブルプロセッサに関係なく，ハードウェア・ソフトウェア協調シミュレーション技術は重要である．

ハードウェア・ソフトウェア協調シミュレータは，ハードウェアの動作を模擬するハードウェアシミュレーションモデル（以下ハードウェアモデル）と，ソフトウェアの動作を模擬する ISS，すなわちプロセッサモデルが協調して動作する（**図 2･15**）．ハードウェアモデルは Verilog などのハードウェア記述言語（HDL）で実装されることが多かったが，HDL シミュレーションはソフトウェア開発用にはスピードが遅く，C 言語などのプログラミング言語で実装されたモデルを使用するようになってきた．そして，HDL より抽象度の高いシステムを記述するための言語がいくつか提案され，現在では SystemC がシステム記述言語として標準になりつつある．

SystemC は C++言語の文法を採用し，ポートチャネルなどのハードウェア構成要素，ス

レッドなどの並列動作，可変長ビットのデータ型などを追加した C++のライブラリである．トランザクションレベルモデリング（TLM）という，RTL より抽象度の高いモデリング手法を提言している．

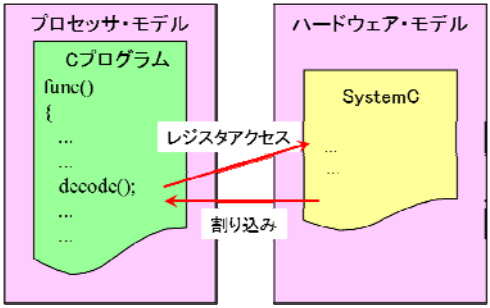


図 2・15　ハードウェア・ソフトウェア協調シミュレーション

2-4-5 ASIP 開発環境の例：ASIP Meister

ASIP Meister はエイシップ・ソリューションズ社の ASIP 開発環境である [5]．プロセッサの仕様を与えることで，プロセッサの HDL 記述とソフトウェア開発ツールを生成する．コンフィギュラブルプロセッサの開発環境と異なり，プロセッサ全体の仕様を与えられるところが特長である．リソースデータベースにハードウェアの IP が登録されており，それらを利用することで設計生産性を高めることができる．またユーザが作成した IP をデータベースに組込むことも可能となっている．

■参考文献

1) 岩戸宏文, 稗田拓路, 田中浩明, 佐藤淳, 坂主圭史, 武内良典, 今井正治, “ASIP 短期開発のための高い拡張性を有するベースプロセッサの提案（アーキテクチャ合成,デザインガイア 2007－VLSI 設計の新しい大地を考える研究会－）,” 情報処理学会研究報告, SLDM, [システム LSI 設計技術], vol.2007, no.114 (20071120), pp.133-138.
2) http://www.semicon.toshiba.co.jp/product/micro/mep/index.html
3) http://www.tensilica.co.jp/
4) 長嶋洋一, “よくわかる組み込みシステムのできるまで,” 日刊工業新聞社, 2005.
5) http://www.asip-solutions.com/index.html

■10 群 － 5 編 － 2 章

2-5 FPGA のアーキテクチャ

（執筆者：笹尾　勤）［2008 年 12 月 受領］

本節では，Field Programmable Gate Array（FPGA）の代表的アーキテクチャと，その応用について述べる．まず，FPGA のアーキテクチャとプログラム用素子に関して概説する．次に，現在最も多く用いられている，テーブル参照形 FPGA に関して，論理ブロックの粒度の問題，配線構造などについて述べる．最後に，FPGA の今後の展開について述べる．

2-5-1 はじめに

現在，種々の FPGA が複数の会社から発売されている．初期の FPGA は，製品開発時のプロトタイプや，少量生産の製品にのみ用いられたが，現在では，民生用大型テレビなどの大量生産品にも使用されるようになっている．その他，通信制御では，頻繁に書き換えを必要とするため，FPGA が古くから利用されている．

FPGA の論理ブロックとしては，トランジスタ，ゲート，PLA（Programmable Logic Array），LUT（Look up table）などがある．また，配線構造も，種々開発されている．プログラム方法も種々のテクノロジーが利用可能である．したがって，多様な FPGA が存在しうる．実際，初期の頃は，多数の企業が種々の構造の FPGA や PLD を開発していた [11]．しかし，現在，FPGA に関してビジネスを続けているのは数社であり，アーキテクチャも整理されてきた．

本節では，まず FPGA における代表的 FPGA アーキテクチャとプログラム用素子に関して概説する．次に，現在最も多く用いられているテーブル参照形アーキテクチャに対して，代表的な設計法，問題点などを述べる．

2-5-2 FPGA アーキテクチャ

本節では，FPGA の四つのアーキテクチャに関して述べる [4), 14)-16)]．

(1)　アイランド形アーキテクチャ

アイランド形（Island-style）アーキテクチャとは，**図 2･16** に示すように，アレイ状に並べた論理ブロックを縦横に走る配線領域で相互接続できるようにしたものである．

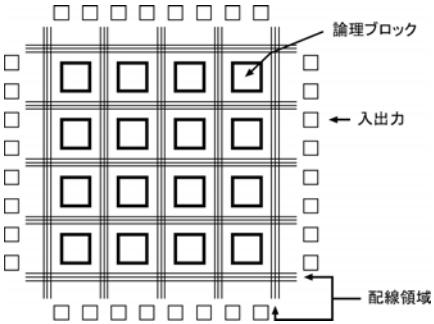


図 2・16　アイランド形アーキテクチャ

XILINX 社や ALTERA 社の FPGA がこの例である．論理素子としては，任意の k 変数関数を実現できる LUT を用いることが多い．また，LUT の入力数としては，$k=4$～6 のものが多い．論理ブロックと配線ブロックは SRAM を用いてプログラムする．繰り返しプログラムでき，システム動作中にもプログラムを変更できるのが大きな特徴である．実際の論理ブロックは**図 2･17** のように複雑で，多出力関数を実現し，フリップフロップを含む．配線接続にはトランスミッションゲートを用いており，接続時の ON 抵抗が高いため，配線の遅延時間が大きい．配線部の遅延が，論理部の遅延よりも大きくなることが多い．通常のゲートアレイに比べ，配線遅延が大きい．大規模 FPGA では，k を大きくしたり，配線部を大きくとって，遅延時間が大きくなり過ぎないように工夫している．

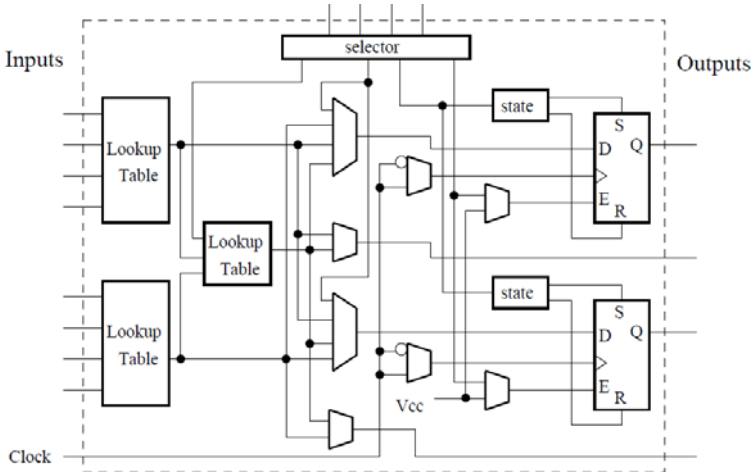


図 2･17　アイランド形アーキテクチャの論理ブロック（XILINX 社 XC 4000）

(2)　行形アーキテクチャ

行形（Row-based）アーキテクチャとは，**図 2･18** に示すように，横に並べた論理ブロックを横に走る配線領域で相互接続できるようにしたものである．

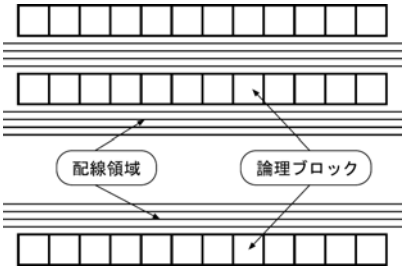


図 2･18　行形アーキテクチャ

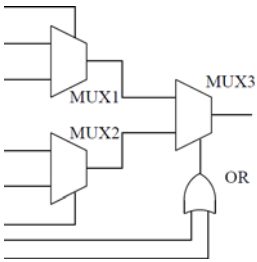


図 2･19　行形アーキテクチャの論理ブロック
（ALTERA 社 ACT-1）

ACTEL 社や QUICK-LOGIC 社の FPGA がこの例である．論理ブロックには**図 2･19** のようなマルチプレクサ（MUX）が入っている．配線接続には低抵抗のアンチフューズを用いており，テーブル参照形 FPGA よりも高速性を狙っている．ただし，プログラムは一度だけ可能であり，変更不能である．

(3) 階層形アーキテクチャ

階層形（Hierarchical-type）アーキテクチャとは，**図 2･20** のように，いくつかの論理ブロックを中央の配線領域で相互接続できるようにしたものである．この FPGA を CPLD（Complex Programmable Logic Device）と呼び，FPGA の範中に入れない人もいる．

ALTERA 社や XILINX 社がこの種の FPGA を発売している．このタイプの FPGA の配線接続は高速で，遅延時間は論理ブロックに比べ無視できる．したがって，遅延時間は論理ブロック段数の整数倍となり，予測が容易である．プログラムは書き換え可能である．

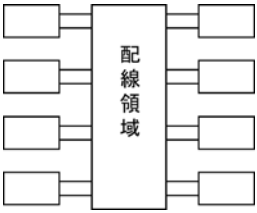


図 2・20　階層形アーキテクチャ

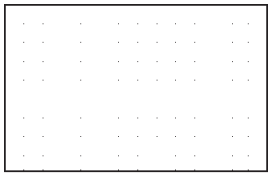


図 2・21　ゲート敷き詰め形アーキテクチャ

(4) ゲート敷き詰め形アーキテクチャ

ゲート敷き詰め形（Sea-of-gates）アーキテクチャの概念図を**図 2･21** に示す．これは，通常のゲートアレイと非常によく似ており，設計法もゲートアレイのものが流用できる．本アーキテクチャでは，設計データが，そのままマスクタイプのゲートアレイの開発に使用できる．CROSSPOINT 社や ATMEL 社がこの種の FPGA を開発していた．

配線接続には低抵抗のアンチフューズを用いており，高速である．ただし，プログラムの変更は不可能である．

2-5-3 プログラム用素子

本節では，FPGA のプログラム用の素子，及びスイッチについて述べる [15]（**表 2･6** 参照）．FPGA のプログラムとは，論理ブロックの内容と，論理ブロック間の配線データを FPGA に書き込むことである．これらのデータをまとめて FPGA の構成データという．

(1) SRAM

SRAM 素子の長所は，何度でもプログラム可能なことである．また，追加プロセスが不要である．最新プロセスが利用可能であり，Moore の法則に従ってデバイスの集積度を確実に改善できる．本素子の短所は，1 ビットを記憶するためにトランジスタを 6 個使用するため，必要なトランジスタ数が多いこと．また，電源を切ると内容が消えてしまう，つまり揮発性である．したがって，電源投入時に外部メモリから構成データを読み込む必要がある．ス

イッチの部分はパストランジスタを用いているため，フューズ形に比べ配線抵抗は大きい．SRAM はテーブル参照形 FPGA で用いられている．

表 2・6　FPGA プログラム用テクノロジー

テクノロジーとプロセス	揮発性	プログラム変更	面　積	スイッチ抵抗〔□〕	寄生容量〔fF〕	追加プロセス数
SRAM MUX パストランジスタ 1.2μCMOS	揮発性	可能	大	0.5〜2 k	10〜20	0
ONO アンチフューズ 1.2μCMOS	不揮発性	不可能	フューズ：小 プログラム：大	300〜500	5	3
非結晶 アンチフューズ 1.2μCMOS	不揮発性	不可能	フューズ：小 プログラム：大	50〜100	1.1〜1.3	3
EPROM 1.2μCMOS	不揮発性	可能 （紫外線）	小	2〜4 k	10〜20	3
EEPROM 1.2μCMOS	不揮発性	可能	EPROM の 2 倍	2〜4 k	10〜20	>5

（J. Rose, A. El Gamal, and A. Sangiovanni-Vincentelli, Proc. IEEE,vol.81, no.7, pp.1013-1029, July 1993 [15].から引用）

(2)　アンチフューズ

アンチフューズとは，非プログラム時には高抵抗値をもつような二端子素子である．素子に 11〜20 V の電圧をかけると，アンチフューズが導通し，低抵抗となる．この変化は不可逆である．アンチフューズとしては，ONO（Oxygen-Nitrogen-Oxygen）構造（**図 2・22**）のものと，非結晶シリコンを用いたものがある．この素子の長所は面積が小さいことである．ただし，プログラム用回路は高電圧と高電流が必要なため面積は大きい．また，素子の抵抗が小さい特長がある．ONO では 300〜500 Ω，非結晶シリコンでは 50〜100 Ωである．また，寄生容量もほかの素子に比べ小さい．また，電源を切ってもプログラムは消えない，つまり不揮発性である．プログラムは一度のみ可能であり，書き換え不可能である．アンチフューズは，マルチプレクサ形 FPGA や，ゲート敷き詰め形 FPGA で用いられている．

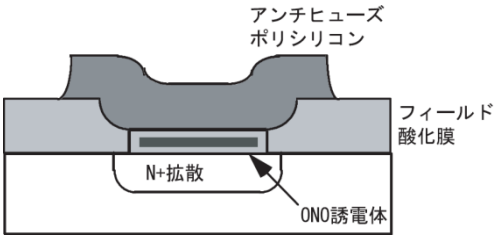


図 2・22　アンチフューズ素子（S. Hauck, Proceedings of the IEEE, vol.86, no.4, pp.615-638, April 1998. [10] から引用）

(3) フローティングゲート

フローティングゲートの技術は，EPROM（Erasable Programmable Read-Only Memory）や EEPROM（Electrically Erasable Programmable Read-Only Memory）で用いられている．EPROM スイッチの回路を**図 2･23** に示す．トランジスタは，二つの制御ゲートを有する．このトランジスタの制御ゲート 1 とドレインの間に高い電圧を印加すると，ゲート 2（フローティングゲート）に電荷が注入され，トランジスタのしきい電圧が上昇しトランジスタは OFF となる．この電荷を除去するにはゲート 2（フローティングゲート）に紫外線を照射すればよい．図 2･23 のような回路にすることで，単にワード線とビット線の間のスイッチとして働くだけではなく，Wired-OR の機能を利用して論理機能をもたせることも可能である．本素子の長所は，CMOS と同様，何度でもプログラム可能なことである．また，電源を切っても内容が消えない特徴があり，外部に余分なメモリを追加する必要はない．短所は，フローティングゲートを作るために余分なプロセスが三つ必要なこと．また ON 抵抗が高いこと（通常の CMOS の 2 倍），更に，負荷に抵抗を用いているため，静止時の消費電力が大きいことである．

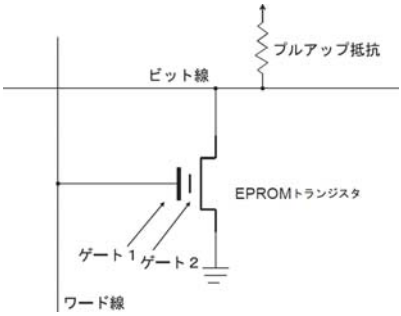


図 2・23　フローティングゲート素子（J. Rose, A. El Gamal, and A. Sangiovanni-Vincentelli, Proc. IEEE, vol.81, no.7, pp.1013-1029, July 1993 [15].から引用）

EEPROM 素子ではゲートの電荷除去に紫外線は不要であり，電気的に電荷を除去可能である．ただし，素子面積は EPROM の 2 倍となる．フローティングゲート素子は階層形 FPGA で用いられている．

2-5-4 テーブル参照形 FPGA

本節では現在最も多く用いられているテーブル参照形 FPGA について述べる．このタイプの FPGA は，図 2･16 のアイランド形アーキテクチャ上に論理素子として SRAM の LUT を用いている．

(1) LUT による論理関数実現法

LUT の入力数を k とし，実現すべき論理関数 f の変数の個数を n とする． $n \leq k$ の場合には，f の真理値表をそのまま LUT に書き込むことにより f を実現できる．$n > k$ の場合には，論理関数 f を一旦 k 入力以下のゲートで実現し，各ゲートを LUT で置換することにより f を実現できる [4), 13)]．また，関数分解 [17)] などの手法を用いて必要な LUT 数を削減することも可

能である．

(2) 論理ブロックの粒度

現在市販中の FPGA では，LUT の入力数は $k = 4 \sim 6$ である．FPGA の面積や遅延時間が最小とするような k の値を求める研究は，多くの研究者の興味を引き付けた．

まず面積に関する最適化であるが，1990 年頃の研究 [14), 12), 5)] では，$k = 4$ のときに面積最小となる結果が出ている．

次に遅延時間に関する最適化であるが，これは論理遅延と配線遅延に分類できる．

- **論理遅延**：k の値を増やすと，LUT 一段当たりの遅延時間は増える．一方，回路の段数は減り，そのための回路全体の論理遅延は減る．
- **配線遅延**：k の値を増やすと，LUT の入力数は $2k$ に比例して増える．また，配線が複雑となり，配線も長くなり，LUT の出力のファンアウトも増える．このため，LUT 一段当たりの配線遅延は大きくなることも実験的にわかっている．

実験によると，LSI の RC 遅延が比較的小さい場合には，全体の遅延は $k = 3 \sim 4$ で最小値をとる．一方，RC 遅延が大きい場合には，全体の遅延は $k = 6 \sim 7$ で最小値をとる．$k = 6$ までは遅延が単調に減少するが，k の値をそれ以上増やしても効果は少ない．1990 年代の FPGA では $k = 4$ のものが多かったが，現在の FPGA では $k = 6$ のものが標準になりつつある．これは微細化により，論理部と配線部の面積の割合や，論理遅延と配線遅延の割合が変化したこと，設計ツールの発展などによる差，設計対象が大規模化したことなどによる．

(3) 論理クラスタ

初期の FPGA では，一つの論理ブロック中に基本論理素子（BLE）が一つだけ入っていた．しかし現在の FPGA では，**図 2･24** のように複数の BLE をクラスタとし，それを図 2･16 の各論理ブロックに埋め込んだアーキテクチャが開発されている．ここで，各クラスタは N 個の BLE を含み，各 BLE は k 入力の LUT とする．また，クラスタの入力数を I とするとき，関係

$$I = \frac{k \times (N+1)}{2}$$

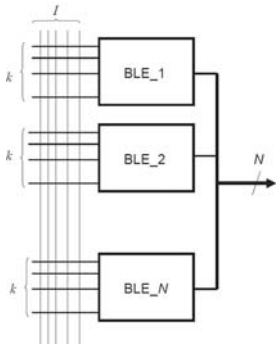


図 2・24　論理クラスタ

が成立するとき，最適になるという研究もあり，ここではそう仮定する．更に，クラスタ内には，ローカルな接続ブロックが用意されており，I 個の入力は，クラスタ内の任意の LUT の入力に接続可能となっているものと仮定する．以上の仮定のもとでの実験によると，次の結論が得られている [1), 2)]．$k=2$〜3 のときよりも $k=4$〜5 のときの方が面積効率がよく，遅延時間も小さい．また，遅延時間は，BLE 数が $N=3$〜10 のとき，LUT の入力数が $k=4$〜6 のとき最小となる．

これらの結論は，特定のベンチマーク関数に対して，特定の論理合成プログラムや配置配線プログラムを適用して得た実験結果によるものである．したがって，ベンチマーク関数やプログラムの進歩とともに，結論も変わる可能性はある．

(4) 配線要素

FPGA 上では，論理ブロックがチップ全体の 10〜20 %を占め，配線要素が残りの 80〜90 %の面積を占める．配線は，**図 2･25** に示すように次の三つの要素から構成されている．

- **接続ブロック**（Connection Block）：論理ブロックとチャネルの配線を接続する．これは，図 2･25 において *C* で示したブロックである．出力部は，パストランジスタ，入力部は，マルチプレクサを用いる．

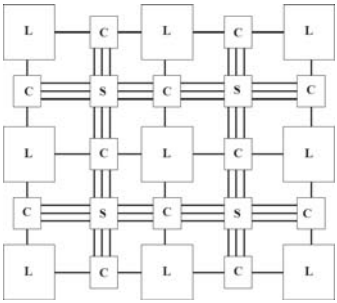


図 2・25　配線要素

- **スイッチブロック**（Switch Block）：**図 2･26** のように，チャネルの垂直線や水平線間の接続を行う．これは，図 2･25 において *S* で示したブロックである．スイッチブロックの各交点は，**図 2･27** で示すように 6 個のパストランジスタで構成されている．

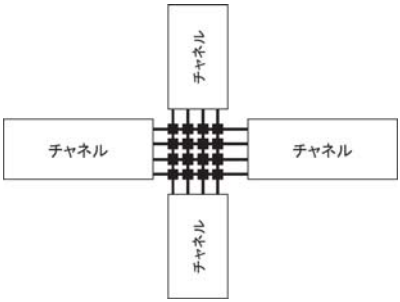


図 2・26　スイッチブロック

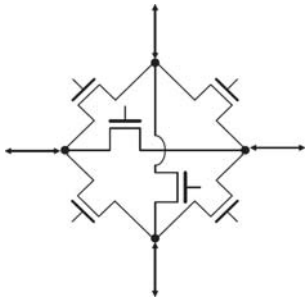


図 2・27　配線用スイッチの詳細

・**配線セグメント**：**図 2･28** で示すように，長さ 1，長さ 2，及び長さ 4 の配線セグメントがある．長い配線セグメントを用いることによって直列に接続されるパストランジスタ数を削減でき，遅延時間も削減できる．このほか，クロック用の配線や電源用の配線もある．

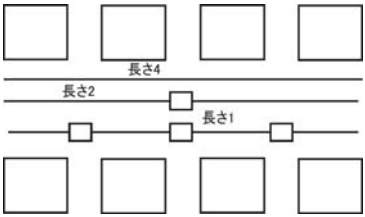


図 2・28　配線用セグメントの構造

2-5-5 その他の構成要素

多くの FPGA では，LUT のほかに，メモリ，プロセッサ，DSP，加算器，乗算器，通信用インタフェースを内蔵しており，1 チップでシステムを実現可能なものも存在する．本節では，そのうち，メモリとプロセッサについて述べる．

(1)　組込みメモリ

多くの FPGA は，LUT のほかに，メモリを組み込んでいる．大きさは，4 K ビットから 18 K ビットのものが多い．更に，大規模なメモリを組み込んでいる FPGA もある．4～6 入力の LUT のほかに組込みメモリを活用した論理設計法も開発されている．LUT の場合，メモリの読み出しは，非同期的に行えるが，組込みメモリの場合には，メモリの読み出しには，クロックが必要なもの，つまり，同期式のものが多い．

(2)　組込みプロセッサ

FPGA のチップ上に一つあるいは複数のマイクロプロセッサ（MPU）を組み込んだ製品が開発されている．これをハードプロセッサ（Hard-macro Processor）という．一方，FPGA 上の LUT などを用いて構成したマイクロプロセッサもある．これをソフトプロセッサコア（Soft Processor Core）という．このうち，Xilinx 社の MicroBlaze と Altera 社の NIOS が有名である．ソフトプロセッサコアの中には，ソースコードが公開されているものもある．

2-5-6 FPGA の今後

FPGA では，デバイスのみならず設計ツールが重要である．各社の FPGA は固有のアーキテクチャを有している．その特徴を生かした設計システムを開発できるのは，その FPGA のアーキテクチャを開発したメーカだけである．高性能な FPGA 設計システムを開発し，それを維持するには相当な投資と年月が必要である．この投資を続けられた企業のみが生き残り，ツールを開発できない企業は淘汰された．

FPGA の専門メーカである XILINX 社（1984 年創業）は，二つの基本特許 [8), 6)] をもとに大きく発展してきた．一方，ALTERA 社（1983 年創業）は，もともとは PLD のメーカであり FPGA に関しては後発メーカであったが，高性能な FPGA 用設計システムと高速アーキテク

チャを武器に FPGA の代表メーカの一つとして健闘している．これらの企業はファブレス，つまりデバイスの製造は他社に任せている．

FPGA は，LSI において，メモリ，マイクロプロセッサに次ぐ地位を占めるようになっている．現在の基本アーキテクチャは既に 20 年以上も使われており，それらの設計資産は膨大である．これらの資産を生かすために FPGA は今後も使用され続けるであろう．

微細化が進むにつれて，一般の論理回路では，素子のばらつきなどの影響で回路設計が困難になっている．一方，FPGA は，規則的構造を有しており，品種当たりの生産個数も十分大きい．したがって，最先端プロセスを活用しても，その開発コストを十分回収可能である．

今後は，リコンフィギャラブルな応用や，ソフトウェア的な使い方も期待される．そのための新しいアーキテクチャや設計アルゴリズムも興味深い [18]．また，構成データの記憶用としてフラッシュメモリや MRAM などの新しいプログラム用素子も開発されている [3), 9)]．

■参考文献

1) E. Ahmed and J. Rose, “The effect of LUT and cluster size on deep-submicron FPGA performance and density,” Proceedings of the 2000 ACM/SIGDA Eighth International Symposium on Field Programmable Gate Arrays, pp.3-12, Feb.10-11, 2000, Monterey, California.
2) E. Ahmed and J. Rose, “The effect of LUT and cluster size on deep-submicron FPGA performance and density,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.12, no.3, pp.288-298, Mar. 2004.
3) http://www.altera.com.
4) S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, “Field Programmable Gate Arrays,” Kluwer Academic Publishers, Boston, 1992.
5) S. Brown, “FPGA architectural research: A Survey,” IEEE Design & Test of Computers, vol.13, no.4, pp.9-15, 1996.
6) W. S. Carter, “Special interconnect for configurable logic array,” U.S. Patent, US4642487, Feb.10. 1987.
7) J. Cong and Y. Ding, “Combinational logic synthesis for LUT based field programmable gate arrays,” ACM Transactions on Design Automation of Electronic Systems (TODAES), vol.1, no.2, pp.145-204, Apr. 1996.
8) R. H. Freeman, “Configurable electrical circuit having configurable logic elements and configurable interconnections,” U.S. Patent, RE34363, Aug. 1993.
9) Y. Guillemenet, L. Torres, G. Sassatelli, N. Bruchon, and I. Hassoune, “A non-volatile run-time fpga using thermally assisted switching MRAMs,” 2008 International Conference on Field Programmable Logic and Applications (FPL-2008), pp.421-426, Sep. 8-10.
10) S. Hauck, “The roles of FPGAs in reprogrammable systems,” Proceedings of the IEEE, vol.86, no.4, pp.615-638, April 1998.
11) J. H. Jenkins, “Designing With FPGAs and CPLDs,” Prentice Hall, Feb. 1994.
12) J. Kouloheris and A. El Gamal, “FPGA performance vs. cell granularity,” Proc. 1991 CICC, pp.6.2.1-6.2.4, May 1991.
13) R. Murgai, R.K. Brayton, and A. Sangiovanni-Vincentelli, “Logic Synthesis for Field-Programmable Gate Arrays,” Kluwer Academic Publishers, Norwell, MA, 1995.
14) J. Rose, R. J. Francis, D. Lewis, and P. Chow, “Architecture of field programmable gate arrays: The effect of logic block functionality on area efficiency,” IEEE J. Solid State Circ, vol.25, no.5, pp.1217-1225, Oct. 1990.
15) J. Rose, A. El Gamal, and A. Sangiovanni-Vincentelli, “Architecture of field-programmable gate arrays,” Proc. IEEE, vol.81, no.7, pp.1013-1029, Jul. 1993.
16) 笹尾　勤, “FPGA の論理設計法,” 情報処理, vol.35, no.6, pp.530-534, Jun. 1994.
17) 笹尾　勤, “論理設計：スイッチング回路理論(第 4 版),” 近代科学社, 2005.
18) T. Sasao, M. Matsuura, and Y. Iguchi, “A cascade realization of multiple-output function for reconfigurable hardware,” International Workshop on Logic and Synthesis (IWLS01), Lake Tahoe, CA, Jun. 12-15, pp.225-230, 2001.

■10 群 － 5 編 － 2 章

2-6 プラットフォーム

（執筆者：荒川文男）［2009 年 6 月 受領］

5 編が対象とする信号処理／通信処理 LSI･プロセッサにとってのプラットフォームはハードウェア（HW），ソフトウェア（SW），あるいは双方を対象とするものまで多岐にわたる．カバーする範囲や目的も様々である．

HW プラットフォームは，標準インタフェースを定義して，これに準拠した標準 Intellectual Property（IP）を作成・蓄積し，標準化された手法で接続して半導体チップを作るための仕組みである．微細化進展の結果，チップは多数の IP を集積した System-on-a-Chip（SoC）あるいは Application Specific Standard Product（ASSP）となっている．そして，多数の IP をすべて自前で新規に作成し，インタフェースをすり合わせてチップを作るという従来方式は，設計期間及びコスト増大を招き破綻しつつあった．更に期間，及びコストを積極的に削減して競争力を高めたいというニーズも高まっていた．こうした背景から，半導体各社は HW プラットフォームを整備し，IP の共通化・再利用性向上を図っている．

SW プラットフォームは，標準 API（Application Program Interface）を定義し，これに合わせたドライバ，OS，コンパイラ，デバッガ，ライブラリ，ファームウェア，ミドルウェアなどを整備し，アプリケーションプログラムの作成及び再利用を容易にするための仕組みである．従来，単一プロセッサに対して整備されてきた SW プラットフォームは，SoC 及び ASSP の発展とともに複雑化し多様化してきた．そして，この結果生じた開発期間，及びコスト増大の抑制，及び削減が必須となった．このため，各社はプラットフォームの集約・共通化やプラットフォーム間の API 互換性を高め，ソフトウェアの再利用性を向上させて，IP として共通化・蓄積を図っている．

本節では，具体的なプラットフォームの例として，パナソニックの Uniphier™，ルネサステクノロジの EXREAL Platform™，NEC エレクトロニクスの EMMA™，東芝のメディア処理プラットフォーム，TI の DaVinci™と OMAP™，及び富士通の組込み用途向けマルチコアプラットフォームについて説明する．

2-6-1 UniPhier™

（執筆者：檜垣信生）［2008 年 12 月 受領］

現在，携帯電話からホーム AV 機器までの幅広いディジタル家電機器において，ユビキタスネットワークを実現する技術が取り込まれ，機器や機能の融合が飛躍的に進展している．事実，ネットワークによる機器制御連携や，SD カードなどの蓄積メディアによるデータ連携が実現されてきている．これらの機能は，従来機器ごとのプラットフォーム上で開発されていたため，同等の機能でありながら個別に開発する必要があり，その結果として多重開発を行っていたという状況がある．更に，機器や機能の融合は，それらに組み込まれるソフトウェアの規模を急激に増加させた．

そこで我々は，これら複数の個別プラットフォームを統合すべく UniPhier を開発した．**図 2･29** に UniPhier の構成を示す．UniPhier は，メディアプロセッサ（UniPhier プロセッサ）を中心としてハードウェアプラットフォームを構築し，その上に商品分野間でソフトウェアの再利用が可能なソフトウェアプラットフォームを構築している．

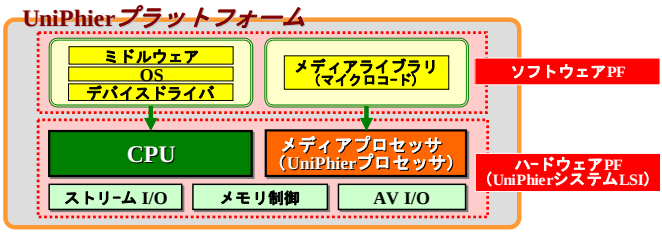


図 2・29　UniPhier の構成

ハードウェアプラットフォームの中核である UniPhier システム LSI は，CPU，UniPhier プロセッサ，メモリ制御ユニット，ストリーム IO，AVIO という 5 つの機能モジュールから構成され，CPU と UniPhier プロセッサを中心としたヘテロジニアスなマルチプロセッサアーキテクチャを採用している．

基幹となるシステムは，CPU を中心としたホストシステムと，UniPhier プロセッサを中心とした AV システムである．メモリシステムは，ユニファイドメモリアーキテクチャを基本とし，ホストシステムと AV システムで共有している．このシステム上に，UniPhier プロセッサメディアライブラリ，OS，デバイスドライバ群，各種フレームワーク（サブセット化可）で構成されるミドルウェア群からなるソフトウェアを搭載し，UniPhier 基本プラットフォームを構成している．

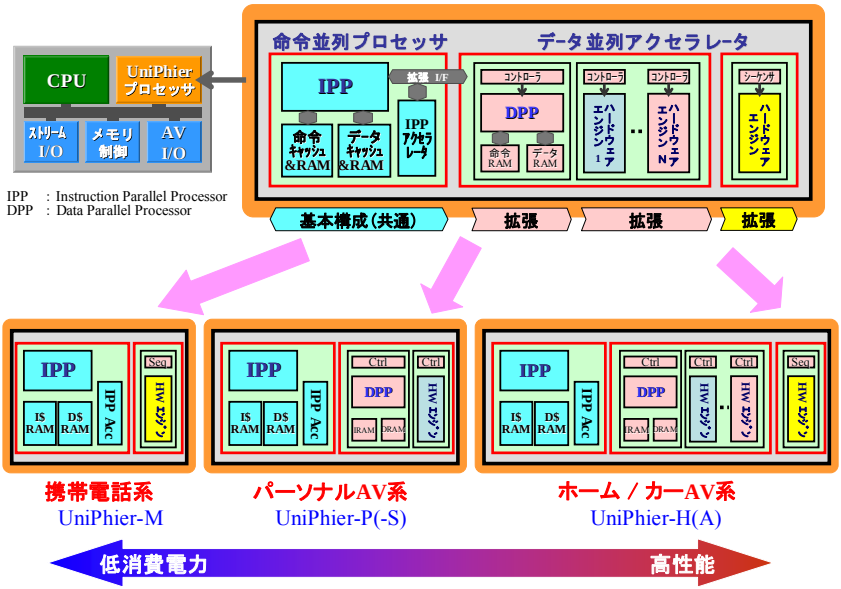


図 2・30　UniPhier プロセッサの構成と展開

AV システムの中核である UniPhier プロセッサにおけるメディア処理は，オーディオ処理，及びメディア処理全体制御に代表される，特定のデータに異なる演算を行う逐次処理と，画素処理のように，大量のデータに同一の演算を行う並列処理に大別される．この性質に基づいて，UniPhier プロセッサでは，命令並列プロセッサ IPP（Instruction Parallel Processor）を核とした逐次的な処理を行うプロセッサモジュールと，データ並列プロセッサ DPP（Data Parallel Processor），及び特定処理向けハードウェアエンジンを搭載したデータ並列アクセラレータモジュールを組み合わせたヘテロジニアスなアーキテクチャを採用した（**図 2･30**）．

UniPhier プロセッサは，IPP を中心とした構造を崩すことなく，対象となるアプリケーションの処理量に合わせてハードウェアの拡張または変更が可能である．UniPhier プロセッサは，ターゲットとする機器に応じたメディア処理を実現するために，ホーム/カーAV 系，パーソナル AV 系，携帯電話系の 3 タイプのシステム LSI に展開している．

携帯電話からホーム AV 機器まで，幅広くディジタル家電に対応し，製品分野を超えた技術の融合や技術の相互利用を加速させるディジタル家電統合プラットフォーム UniPhier を開発し，商品分野間でのソフトウェア資産とハードウェア資産の相互活用が可能となり，開発の効率を向上させ，ディジタル家電の飛躍的な展開を実現した．今後，更なる展開を図ると共に，ユーザインタフェース改善など次世代への更なる進化を行っていく．

2-6-2 EXREAL Platform™

（執筆者：服部俊洋）［2009 年 3 月 受領］

EXREAL Platform™（EXcellent Reliability Efficiency Agility Link Platform）は先端組込みシステムのハードウェア，ソフトウェアの構築を対象として 2004 年に発表され，発表後も継続的に拡張を行っているマザープラットフォームである [1), 2), 3)].

EXREAL Platform™ を構成する技術の全体像を**図 2･31** に示す．EXREAL Platform™ は大別すると，コンポーネントデバイスとインターコネクト技術からなる．コンポーネントデバイスは，ハードウェア部品（ハードウェア IP）とソフトウェア部品（ソフトウェア IP）を意味する．

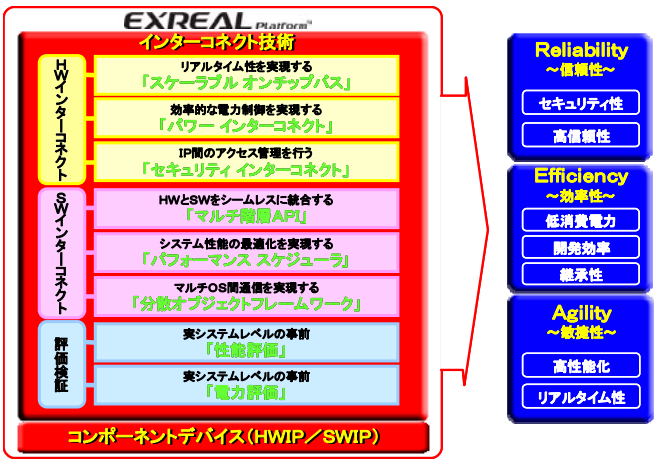


図 2･31 EXREAL Platform™ を構成する技術

インターコネクト技術は，ハードウェア（HW）インターコネクト技術，ソフトウェア（SW）インターコネクト技術，評価検証技術で構成される．HW インターコネクト技術は，ハードウェア IP コア間を接続する技術である．信号転送や電力制御，クロック周波数制御，アクセス管理などの機能と接続・検証容易化技術を備える．SW インターコネクト技術は，ソフトウェア IP であるデバイスドライバや OS，ミドルウェアなどを適切に接続する技術である．ソフトウェアの各階層に応じたインタフェース（階層 API），電力と周波数の制御機能，CPU コア数をソフトウェアから見えなくするフレームワークなどを含む．

評価検証技術は，システムレベルで性能，機能を（事前検討段階を含む）開発のそれぞれのフェーズで評価検証する技術である．

(1) HW インターコネクト技術

図 2･32 に，HW インターコネクト技術の中核となるバスプラットフォームアプローチを記載する．HW インターコネクトとは，CPU，及びマルチコア IP や VPU（Video Processing Unit），3D グラフィックスアクセラレータ，DDR I/F，USB 2.0，認証，暗号などの各種ハードウェア IP をつなぐ技術となる．キーとなるのは，それらハードウェア IP を物理的に接続するオンチップバスである．

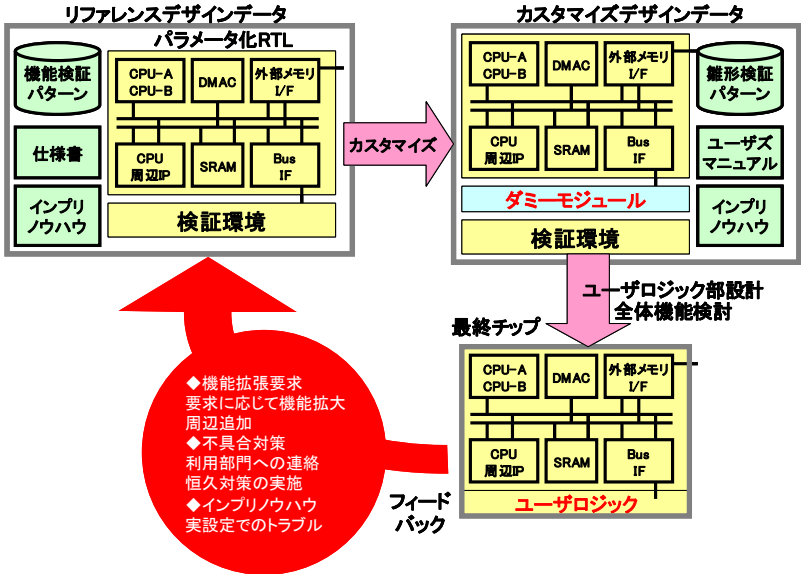


図 2・32　EXREAL Platform™ でのオンチップバスプラットフォームアプローチ

EXREAL Platform™ のオンチップバスは，転送速度やシステム構成などの要求仕様に応じて構成を変えられるようになっている．この特徴をスケーラブルオンチップバスと呼んでいる．スケーラブルオンチップバスの設計段階で，論理設計者が条件（転送速度・搭載 IP 数）を入力するとハードウェア記述（RTL 記述）が自動的に生成される [4]．

HW インターコネクトの中では，スケーラブルオンチップバスのほかに，次の二つの技術も提供する．オンチップバスパワーインターコネクトは，電力制御機能とリソースマネージャによる低電力制御のための技術であり，効率的な電力制御を実現できる．セキュリティインターコネクトは，IP 間のアクセス管理を行うアクセス制御機能によって高信頼性化を実現できる技術である．

(2) SW インターコネクト技術

ソフトウェア開発では，システムメーカや半導体ベンダ，ソフトウェアベンダ間の複数の開発を総合して完成するのが一般的である．その際，開発分担を明確にすること，及び，開発済みのソフトウェア資産を効率よく再利用することがポイントとなる．

EXREAL Platform™ では，ソフトウェア階層間のインタフェース（API）を 3 通りに定義した（**図 2･33**）．アプリケーションに近い方から，アプリケーションレベル API，ミドルウェアレベル API，デバイスレベル API である．これによって，システムのカスタマイズの範囲をアプリケーションプログラムのみに絞ったり，ミドルウェアの階層にも手をいれたりといった，異なる要求に対応することができる．

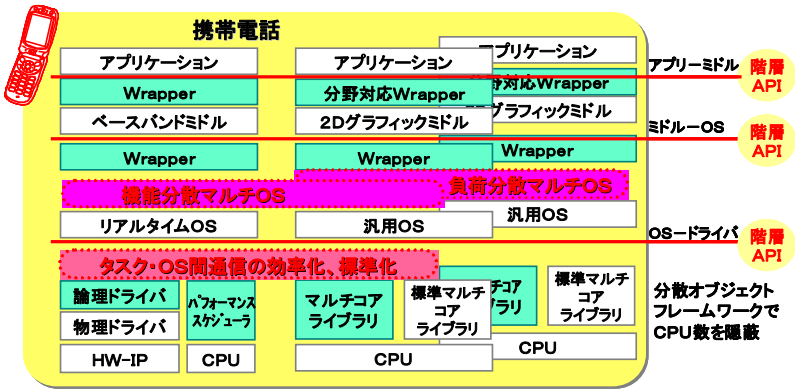


図 2･33 SW インターコネクト

また，製品分野ごとに標準的な API が異なっていたり，主流の OS が異なっていることが多い．そこで，ラッパー（Wrapper）によって API や OS の違いを吸収するようにした．

また，近年の半導体集積度の向上にともない，複数の CPU コアを一つの SoC に搭載するケースが増加している．ここで，ソフトウェアの階層化が不十分であると，新規の SoC 開発で CPU コアを増やした場合にアプリケーションを書き直す必要性が発生する．EXREAL Platform™ では，マルチコアライブラリと呼ぶソフトウェアでこの問題に対処した．CPU コアと動作する OS の数が増えても，アプリケーションからは仮想的に単体の OS が見えるようにし，CPU コアの数を見えなくする．マルチコアライブラリではどの機能をどの OS が処理するかというマッピング情報に基づき処理の依頼先 OS を特定し割り当てる．

(3) 評価検証技術

システム開発の中の最上流は，システムレベルの設計である．システムが担当する機能の概略を設計し，ハードウェアで処理する部分とソフトウェアで処理する部分に分割する．このシステムレベル設計の段階で機能や性能などを評価，検証することが重要である．

EXREAL Platform™ では評価検証技術と呼ぶ，システムレベルで機能を検証し，性能を予測する仕組みを備えた．システム設計に使われる言語である SystemC を使い，性能を評価する環境を構築した（**図 2･34**）．

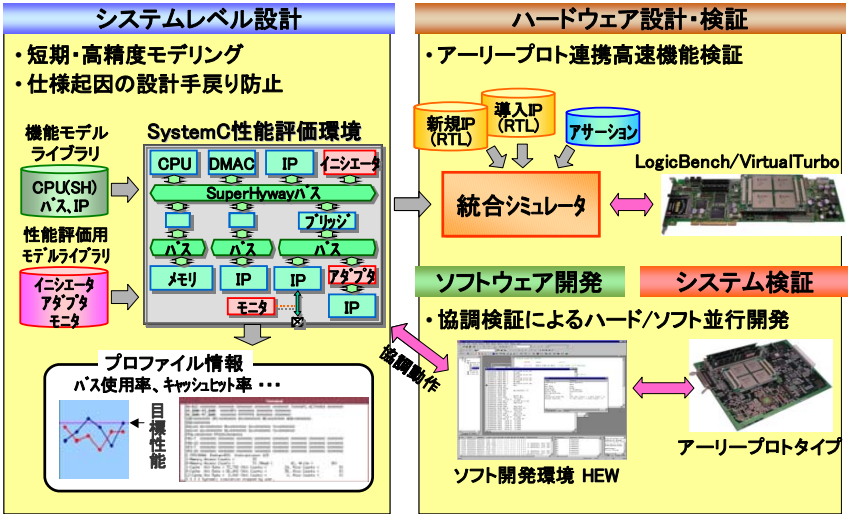


図 2・34　評価検証技術での評価環境

機能の検証では，CPU やバス，IP コアなどの機能モデルを SystemC の性能評価環境に組み入れることによって実現する．

ハードウェア設計では，論理検証を加速するために，FPGA ボードにハードウェア（論理回路）を搭載して機能検証を行う．ソフトウェア開発でも FPGA ボードとターゲットシステムを組み合わせたものをハードウェアとして使い，ハードウェア設計と並行してソフトウェアを開発することによって設計期間を短縮する．

SystemC の性能評価環境では更に，消費電力をシステム設計の初期段階で見積もれるようにする．この環境では，電力評価用の各コンポーネントのモデルライブラリを用い設計初期に消費電力を予測することで，設計の手直しが発生する可能性をなくす．

■参考文献

1) T. Hattori, “EXREAL Platform: SOC Design Challenges for Embedded Systems,” Cool Chips X, 2007.
2) ルネサステクノロジ, “特集 すべてをつなぐ革新的なプラットフォーム EXREAL Platform™,” EDGE, vol.12, pp.1-10, 2006.
3) ルネサステクノロジ, “特集 発想力の差が先端技術を加速する ルネサス EXREAL Platform™,” EDGE,

Vol.21, pp.1-11, 2008.
4) 辻本芳孝, 野々村到, 吉田　裕, “パケットルータジェネレータの開発,” 電子情報通信学会 集積回路研究会 第 11 回システム LSI ワークショップ, 2007.

2-6-3 EMMA ™

（執筆者：境　則彰）［2009 年 1 月 受領］

ディジタル AV 機器向けの SoC はプラットフォーム型の内部アーキテクチャを採るものが多く，適用分野もディジタル TV，STB，DVD，Blu-ray Disc（BD）など多岐な分野に渡っている．プラットフォーム型のアーキテクチャが採用される目的としては，多種・多様な適用分野に対応するような SoC 製品群を高品質，かつ効率よく開発できるという点にある．このようなプラットフォームの一例として，“EMMA™” と呼ばれる SoC アーキテクチャを例に説明する．

EMMA™は Enhanced Multi-Media Architecture という意味で，MPEG 圧縮／伸張などメディア処理を効率よく処理するために開発された．内部はコマンドバス（C-Bus），メモリバス（M-Bus）という 2 種類のバスと，それらに接続されるメディア処理用の内部ユニット，及び CPU，ペリフェラルにより構成されている（**図 2･35**）

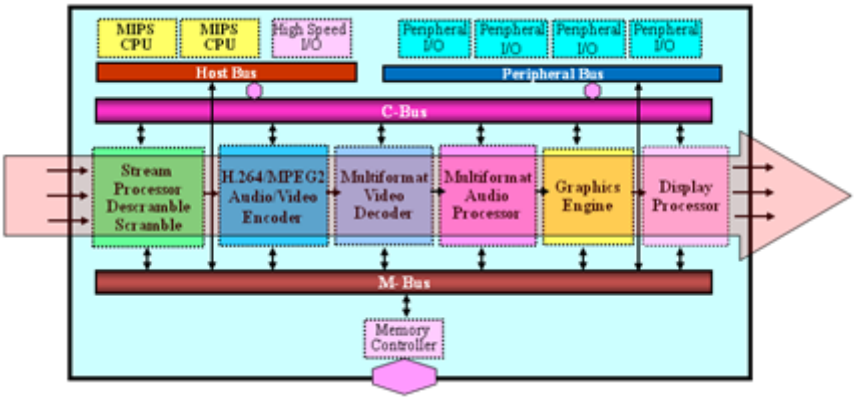


図 2･35　EMMA の内部構造

(1)　バス構造とメモリコントローラ

SoC の内部はコマンドバス（C-Bus），メモリバス（M-Bus）という 2 種類のバスにより構成されている．CPU はコマンドバスによりメディア処理用の内部ユニットのレジスタにアクセスできる．レジスタアクセスにより，各ユニットの動作パラメータや処理モードの設定，コマンド入力やステータス読出しが可能となる．一方，メモリバスにはメモリコントローラを経由して外部に SDRAM などのメモリが接続される．SoC の適用分野・用途に応じて必要とされるメモリ容量，帯域が異なるため，メモリバスのバス幅，バスクロック速度，及び，外部メモリは SoC に最適なものが選択される．外部メモリとしては，SDR，DDR，DDR2 などの SDRAM や RDRAM の中から選択される．メモリコントローラは，このような外部メモリの種類に合わせて制御する階層のほか，内部ユニットなどが必要とするメモリ帯域，容量を考慮して優先度制御するような外部メモリの種類に依存しない階層で構成される．優先度

制御はSoCシステム設計の段階で想定されたメモリ帯域が保障できるようなQoS制御を特徴としている．また，CPU のようなキャッシュメモリを備えるユニットでは帯域保障よりアクセスレイテンシの短縮が重要である．メモリコントローラは，メモリ帯域を保障することと，CPU アクセスのレイテンシの短縮を両立できるように設計されている．

(2) メディア処理ユニット

メディア処理ユニットは，ディジタル AV の信号処理を担当する部分である．暗号化処理，DeMux 処理を含むストリーム処理ユニット，ビデオ・オーディオのコーデック処理ユニット，2 次元グラフィクス処理ユニット，複数表示プレーンを含む表示処理ユニットなどを含む．各ユニットは SoC の適用分野により要求される機能，性能に違いがある．その違いに柔軟に対応するため，プログラマブル化，ユニット内のモジュラ化などの対応がとられている．プログラマブル化を実現するためにC言語で記述されたリコンフィギュアブルプロセッサが活用されている．このプロセッサには適用するユニットに必要な処理を効率よく実効できる命令の追加が可能である．例えばストリーム処理ユニットでは，MPEG-TS パケットの処理に適した命令を備えたプロセッサを内蔵し，グラフィクス処理ユニットでは字幕など文字の展開に適した命令が追加されている．またユニット内部もモジュラ化が進んでおり，ビデオ表示用プレーンやグラフィクス表示用のプレーンなどでプレーンの解像度，プレーン数の組合せを用意に実現できるような構造を実現している．ビデオデコーダやオーディオデコーダのコーデックの追加に関してもモジュラ化が考慮されており，必要な機能，性能に合わせた構成が実現可能である．

(3) CPU とペリフェラル

SoC において CPU は重要な構成要素であるが，SoC に必要とされる処理に応じて最適な構成が選択される．メディア処理ユニットのハードウェアを制御したり，ペリフェラルを使用した入出力制御にはリアルタイム性が重視される．一方，ディジタル TV におけるデータ放送処理や BD の BD-J 処理などは，HD コンテンツのメディア処理やスクリプト処理や JavaVM 処理などコンピューティング処理が重視される．また，ネットワーク対応などのプロトコルスタックは Linux を活用すると環境面で充実している．このようなことから適用分野に応じて下記の 2 種類の構成が準備されている．

①アプリケーションの負荷が比較的高い用途（ディジタル TV・BD など）

アプリ用 CPU ＝ 高性能 CPU ＋ LinuxOS

制御用 CPU ＝ 高効率 CPU ＋ RTOS

②アプリケーションの負荷が比較的低い用途（STB・DVD など）

アプリ／制御兼用 CPU ＝ 高効率 CPU ＋ RTOS

高性能 CPU の例としては VR 5500 などがある．VR 5500 はスーパースカラ型 CPU であり，投機的命令実行や分岐予測など様々な高速化機構が採用されている．一方，高能率 CPU には典型的な 5 段パイプライン RISC CPU などが採用される．

ペリフェラルには SATA，USB，Ethernet などの比較的高速，高機能なものから，シリアルインタフェースのような比較的低速，単純なものまであり，それぞれの特性に合わせてペリフェラルバスに集約される．ペリフェラルバスはコマンドバス（C-Bus），メモリバス(M-Bus)

にバスブリッジを介して接続される．

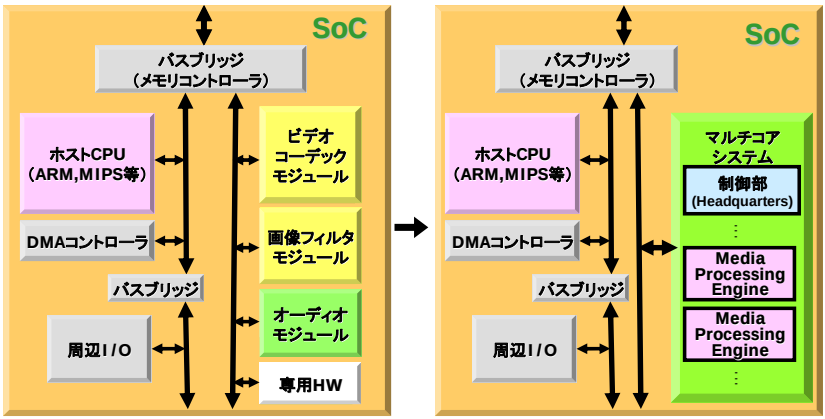
以上の説明のようにプラットフォームを活用することで，(1) バス構造とメモリコントローラを骨格とし各種適用分野に応じて，(2) メディア処理ユニット，(3) CPU とペリフェラル，を組み込むことで，各種適用分野に適した SoC を効率よく開発することが可能となる．

2-6-4 マルチコアによるメディア処理プラットフォーム （執筆者：田辺　淳）[2008 年 12 月 受領]

本項では，東芝セミコンダクター社の開発した，キャッシュメモリベースのマルチコアシステムを使ったメディア処理プラットフォームについて述べる．2008 年に本プラットフォームをベースとした 8 プロセッサ構成の LSI [1), 2)] が発表されており，携帯電話機向けアプリケーションプロセッサ LSI などへの適用が考えられている．

(1)　プラットフォームの特徴

従来の組込み向け SoC の例を**図 2･36** (a) に示す．組込み向けのマルチメディア処理，特にリアルタイムの動画像圧縮伸張処理などは，必要な演算処理性能に対するコストや消費電力の観点から，用途に応じたハードワイヤド型のアクセラレータを組み合わせる方式がとられてきた．しかし，近年では携帯電話機などにおいて性能向上とサービスの拡充が急速に進んでおり，ハードワイヤド型の IP が次の世代では性能不足になって流用できないケースや，LSI の仕様策定後に新たな機能を追加する必要が出てくるなど，従来のようなハードワイヤド型 IP ベースの開発手法をとることが難しくなってきた．



(a) 専用モジュールベースの SoC　　(b) マルチコアシステムベースの SoC

図 2･36　マルチコアシステムベースの SoC

そこで，本プラットフォームでは，ハードワイヤド型の IP ではなく，図 2･36 (b) のようなマルチコアシステムで動作するソフトウェアによって，幅広い製品分野や多様な機能要求に対応することを目指している．本プラットフォームの最も大きな特徴としては，プロセッサコア数を仮想化したソフトウェア実行環境上でファームウェアを動作させる点がある．こ

れにより，プロセッサコア数が異なる複数の製品で同じファームウェアを動作させることが可能になるほか，次の製品で大きな画面解像度に対応するときなどには，プロセッサコア数を増やすことによって，スケーラブルな性能向上を実現することができる．また，メディア処理用のシンプルなプロセッサを多数並べることにより，小面積かつ低消費電力なシステムを実現していることや，ファームウェアを変更することなく様々なメモリ構成に対応できるように，スクラッチパッド型のローカルメモリではなく，キャッシュベースのメモリシステムを採用していることも大きな特徴である．

(2) ハードウェアの構成

本プラットフォームのマルチコアシステム構成を**図 2･37** に示す．基本的なアーキテクチャとしては，①Headquarters と呼ばれる全体をコントロールする制御用プロセッサ，②MPE（Media Processing Engine）と呼ばれるメディアプロセッサ，③全コアで共有する 2 次キャッシュメモリシステムから構成される，キャッシュベースのマルチコアシステムである．Headquarters と MPE は，共に Media embedded Processor（MeP）[3] と呼ばれるコンフィグラブルプロセッサをベースにしており，どちらも 1 次キャッシュメモリをもつ．

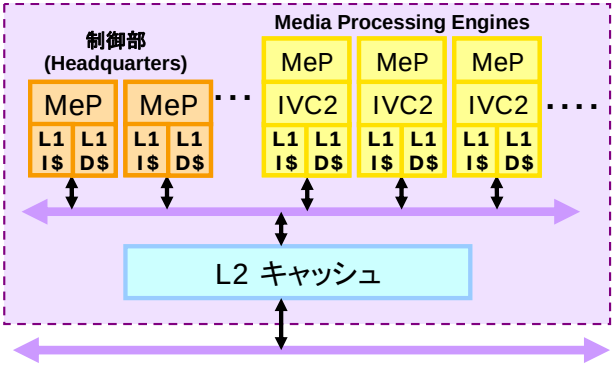


図 2･37　ハードウェア構成

Headquarters はシンプルなシングルスケーラの RISC プロセッサであるが，MPE はメディア処理性能を高めるために，MeP に SIMD/VLIW 型のコプロセッサ「IVC 2」を接続している．MeP と IVC 2 は密に結合して，最大 3 並列の VLIW プロセッサとして動作し，最大 8 並列の SIMD 命令を実行することにより，通常の RISC プロセッサに対してメディア性能を高めている．また，外部バスとの間に共有 2 次キャッシュメモリをもつことにより，システムごとに異なる外部バスの影響を減らし，システムレベルでの性能保証を行いやすくしている．

本プラットフォームでは，基本構成のパラメータ（MPE 数，1 次キャッシュメモリ容量，2 次キャッシュメモリ容量など）を変更することで，同一のファームウェアを利用しながら様々な要求性能の機器に展開することを想定している．一つの MPE の粒度が大きいと，最小構成での面積や消費電力が大きくなり，携帯電話機のローエンド品などへの適用が困難になる．このため，MPE は単体での性能よりも電力効率や面積効率を重視した設計になっ

ている [4].

(3) ソフトウェア並列化手法

本プラットフォームで動作するファームウェアでは，並列化手法としてマルチスレッド型並列化を使用している．ソフトウェア実行環境モデルを**図 2･38** に示す．各プログラムは複数のタスクをもち，各タスクの内部で Headquarter が複数のスレッドを生成する．生成されたスレッドは，カーネルが提供するスケジューラによって各 MPE へ割り当てられ，MPE 内の演算リソースを使って処理が行われる．タスクからは，具体的に幾つの MPE があるかは見えず，MPE の数に依存することなく動作する．本プラットフォームでは，これらの処理は V-Kernel と呼ぶ軽量なカーネルを使って実現される．

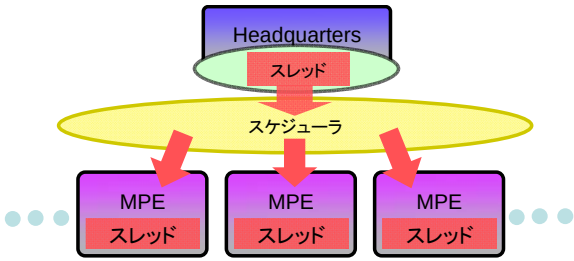


図 2･38　ソフトウェア実行環境

このように，本プラットフォームでは V-Kernel が提供するスケジューラを介することで，MPE の演算リソースを抽象化し，同一バイナリのファームウェアを MPE 数やキャッシュメモリ容量の異なるシステム上で再利用することを可能としている．

■参考文献

1) S. Nomura, F. Tachibana, T. Fujita, C. K. Teh, H. Usui, F. Yamane, Y. Miyamoto, C. Kumtornkittikul, H. Hara, T. Yamashita, et al., “A 9.7mW AAC-decoding, 620mW H.264 720p 60fps decoding, 8-core media processor with embedded forward-body-biasing and power-gating circuit in 65nm CMOS technology,” IEEE International Solid-State Circuit Conference, pp.262-263, 2008.
2) Y. Ueda, N. Nonogaki, T. Terazawa, T. Kodaka, T. Mori, K. Morita, H. Arakida, T. Miura, Y. Miyamoto, Y. Okuda, T. Kizu, Y. Tsuboi, “A Power, Performance Scalable Eight-Cores Media Processor for Mobile Multimedia Applications,” IEEE Asian Solid-State Circuits Conference, pp.13-16, 2008.
3) T. Miyamori, “A configurable and extensible media processor,” Embedded Processor Forum, 2002.
4) S. Hosoda, M. Uchiyama, J. Tanabe, K. Yasufuku, T. Tamai, T. Matsumoto, T. Miyamori, and M. Nakagawa, “A Low-power Mobile Multimedia Processor for Scalable Multi-core System,” Proc. of IEEE Cool Chips XI: Symposium on Low-Power and High-Speed Chips 2008, Apr. 2008.

2-6-5 DaVinci™と OMAP™

（執筆者：佐藤登志）［2009 年 7 月 受領］

(1) はじめに

テキサス・インスツルメンツ（TI）は，1982 年に世界初のプログラマブル汎用DSP[*1] である「TMS 320 C10」を発表してから今日まで，継続性を重視したDSPの開発を続けている．1980 年代後半にCMOSプロセス技術を採用，アドレス生成機能を追加し，32 ビット精度の固定小数点DSPを開発している．1990 年代には 100 MHzを超えるクロックを実現し，2 次キャッシュを搭載，浮動小数点DSPを開発した．1990 年代後半にはVLIW[*2] やデュアルデータバスアーキテクチャを開発・採用している．2000 年以降も 1 GHz超のクロック速度を実現し，デュアルコアアーキテクチャや高速外部インタフェースの採用など技術革新に取り組んでいる．

一方で，携帯電話の高機能化に端を発したモバイル機器のマルチメディア対応の要求には，DSPに専用サブシステムを追加したSoC[*3] を開発している．最新デバイスは，高性能化と同時に設計仕様も高度化，複雑化していることから，各種ソフトウェア／アルゴリズムやフレームワーク，ツールを含む開発環境をデバイスと共に提供することで，最終製品の開発を容易にしている．本項ではTIの代表的なプラットフォームとしてDaVinci ™ とOMAP ™ を紹介し，最後に今後の展望についても言及することにする．

(2) TI のプラットフォームについて

TI のプラットフォームの最大の特徴は，開発資産の再利用にある．同一プラットフォーム内のプロセッサ間にとどまらず，異なるプラットフォームにおいても，同一コアを採用するプロセッサ間であれば同じく再利用が可能である．これはプロセッサがプログラマブルアーキテクチャを採用しており，TI の標準規格に準拠した開発ツールやソフトウェア／アルゴリズムが TI のみならずサードパーティからも提供されているためである．プラットフォームの選択の自由度が大きいことから，要求仕様が大きく乖離する製品や，全く異なるアプリケーションにおいても開発資産の再利用や流用が可能となり，開発にかかる負荷を低減できる．

TI ではディジタル信号処理のプラットフォームを組込みプロセッサ（Embedded Processors）と総称し，DSP のプラットフォームと共にマイクロプロセッサやマイクロコントローラを包括している．DSP のプラットフォームには C5000™ 低消費電力 DSP や C6000™ 高性能 DSP と，SoC のプラットフォームである DaVinci ディジタルメディアプロセッサと OMAP アプリケーションプロセッサが含まれる．DaVinci は高度なディジタルビデオイメージング処理機能を提供する製品群であり，OMAP はポータブル機器の低消費電力で最新アプリケーションやグラフィックス，ネットワーク機能を実現する製品群である．次に SoC のプラットフォームを紹介する．

(3) Vinci プラットフォーム

Vinci プラットフォームは高度なディジタルビデオイメージング処理機能を提供するプラットフォームである．プロセッサを選択することで，入力フォーマットや画像サイズを問わず，最大で 1080 p までの HD 画像処理や各種コーデックに対応する．また圧縮された動画

*1 DSP：ディジタルシグナルプロセッサのこと．TI ではディジタルシグナルプロセッシングと表記．
*2 VLIW：長命令語（Very Long Instructing Word）のこと．
*3 SoC：システムオンチップ．

データをデコードせずに別の形式に変換したり，解像度などの異なるデータに変換することも可能である．例えば MPEG-2 から H.264 や，H.264 における高いビットレートから低いビットレートへの変換である．また，多チャンネル入力の処理も可能である．サブシステム／コプロセッサによる高機能性とプログラマブルアーキテクチャによる拡張性を備えており，TI の提供するライブラリや独自アルゴリズムの組込みが容易なことから，アプリケーションを問わず採用可能である．

DaVinci プラットフォームはディジタルメディアプロセッサと，ソフトウェア開発ツール，及びアプリケーションソフトウェア／アルゴリズムで構成される．

ディジタルメディアプロセッサはTIのTMS 320C64x+ ™ 32 ビット固定小数点DSPコア [*1] のサブシステムと 32 ビットRISC CPUサブシステム，ビデオ処理サブシステムのほか，VICPやMJCP，HDVICPなど製品により呼称が異なるが製品仕様に合わせて最適化されたディジタルビデオイメージング処理専用のサブシステム／コプロセッサが搭載される．同じく製品仕様に合わせて最適化されたペリフェラルとともに，これらの組合せで性能，機能が異なる製品を提供する．**表 2･7** 特徴的なプロセッサを例示する．

表 2・7　aVinci　ディジタルメディアプロセッサの製品例

	DM6467	DM648	DM6446	DM365
32ビットRISC プロセッサ	ARM926 EJ-S 360MHz	-	ARM926 EJ-S 300MHz	ARM926 EJ-S 300MHz
32ビット固定小数点 DSP	TMS320C64x+ 720MHz	TMS320C64x+ 900MHz	TMS320C64x+ 600MHz	-
HDビデオ（例）	1080p 30fps H.264 1 ch	CIF H.264 10ch	720p 15fps 1 ch	720p H.264 30fps
ビデオポート	入力1 系統、出力1 系統 (8bit それぞれ2系統)	入出力5系統 (8bit 10系統)	入力1 系統 出力1 系統	入力1 系統 出力1 系統
マルチメディア アクセラレータ	HD VICPx2	SD VICP	SD VICP / OSD	HD VICP / OSD
EMAC	GMII	2x SGMII	MII	MII
HDDインターフェース	PATA	-	PATA	-
PCI	33MHz	66MHz	-	-

表 2･7 中から，DM 6446 を例に設計仕様を説明する．

このディジタルメディアプロセッサは，CPU サブシステムと DSP サブシステム，ビデオ処理サブシステム，ビデオイメージングコプロセッサ（VICP）と各種インタフェースで構成される（**図 2･39**）．

CPU サブシステムは 32 ビット RISC プロセッサで，全体の制御を主な目的としており，パイプライン処理を前提とした構成で 16×32 ビットの乗算器と密結合メモリ，命令キャッシュ，データキャッシュ，内蔵 RAM，内蔵 ROM，メモリ管理ユニットを備え，専用 32 ビット／16 ビット命令により動作する．

*[1] TMS320C64x+：TI の代表的なハイパフォーマンス DPS コアのこと，C64x+と略称．登録商標．

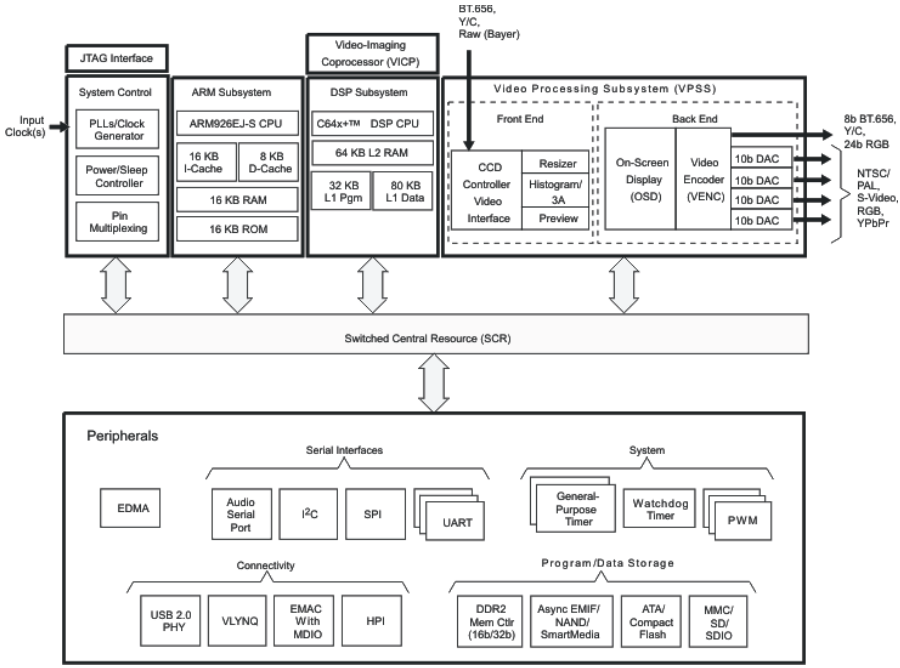


図 2・39　DM6446　ブロック図

DSP サブシステムは TI の TMS 320C64x+コアの 32 ビット固定小数点プログラマブル汎用 DSP で，本プロセッサではディジタルメディアアプリケーションを含む各種ディジタル信号処理を行う．**図 2･40** に構成を示す．

C64x+コアは 8 個の機能ユニットと 2 個の 32 ビットレジスタファイル，同じく 2 個のデータパスをもち，第 2 世代 VLIW アーキテクチャを採用している．8 個の機能ユニットは 6 個の論理演算ユニットと 2 個の乗算器で構成され，各ユニットが 1 サイクルで 1 命令を独立処理している．VLIW アーキテクチャにより単純なハードウェア構成で複数機能ユニット分の命令を同時に取り込み，実行している．また，スモールインストラクションバッファの採用により，コードサイズを抑えながら使用頻度の高い命令語を 32 ビットから 16 ビットとしてレジスタファイルを有効活用している．プログラムが命令の実行中にバグを自身で排除する工夫もされている．

ビデオ処理サブシステムは，イメージセンサやビデオデコーダなどからの外部入力信号をフロントエンドで画像処理し，バックエンドでアナログテレビやLCDパネル，HDテレビ用ビデオエンコーダなどへ出力する．VICPはSIMDアーキテクチャのコプロセッサであり，1 サイクルで 8 命令の同時処理が可能である．画像処理に使われるCFA補間フィルタや，マトリックス演算，色空間変換，FFT，DCTなど各種演算に適している．ペリフェラルには外部メモリインタフェースやDDR 2 メモリコントローラのほか，シリアルインタフェースとしてUARTやチップセレクト付きのシリアルペリフェラルインタフェース（SPI），マスタとスレ

ーブの両モードに対応するI2C [1]，オーディオシリアルポート（ASP）などが用意されている．16 ビットパラレル通信のホストポートインタフェース（HPI）やPHY内蔵のUSB2.0HS，高速シリアル通信インタフェース「VLYNQ」，最高 100 Mb/sのイーサネットマック，PWM [2] 出力や，汎用タイマ，ウォッチドックタイマも備えている．

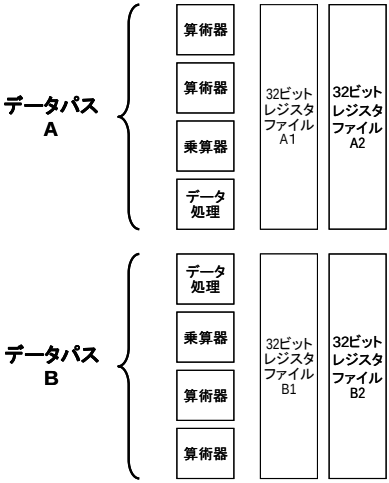


図 2・40　TMS 320C64x+コア　ブロック図

(4)　OMAP プラットフォーム

OMAP プラットフォームは，消費電力が大きく制限されたポータブル環境において最先端のアプリケーションやグラフィックス，ネットワーク機能を提供するプラットフォームである．TI の低消費電力プロセステクノロジーを採用しているアプリケーションプロセッサはサブシステムを統合するアーキテクチャを採用し，タスクによりサブシステムレベルでパフォーマンスと消費電力を最適化している．また，TI 独自のパワーマネージメント技術であるSmartReflex™テクノロジーを搭載し，デバイスの使用状況，動作モード，プロセステクノロジー，温度変化に応じて電圧，周波数，消費電力を動的に制御可能である．SmartReflex ™ テクノロジーの効率を最大限に高める電源制御プロセッサも用意している．いずれも携帯電話端末の高機能において採用，実証された技術である．

このプログラマブルアーキテクチャを採用しているアプリケーションプロセッサは各種ハイレベルOS [*3] に対応しており，ハードウェアを意識することなくアプリケーション層における独自アルゴリズムの新規開発や既存資産の移植が可能である．

ナビゲーションデバイスやメディアプレーヤ，ゲーム端末など民生分野に加え，計測機器や大型装置のユーザインタフェースなど産業分野でも活用できる．

OMAP プラットフォームはアプリケーションプロセッサと，ソフトウェア開発ツール，及

*1 I2C：インターインテグレーテッドサーキット（I2C Bus™）．
*2 PWM：パルスウィデュスモデュレータ．
*3 ハイレベル OS：Linux や Windows CE など．

びアプリケーションソフトウェア／アルゴリズムで構成される．

アプリケーションプロセッサは RISC プロセッサと各種ペリフェラルを標準搭載し，DSP サブシステムや, 2D/3D グラフィックスアクセラレータ, マルチメディアサブシステム（IVA 2.2）との組合せで製品群を構成する．製品群を**表 2･8** 示す．

表 2・8OMAP 35x　製品群

	OMAP3530	OMAP3525	OMAP3515	OMAP3503
32ビット RISC プロセッサ	ARM Coretex™-A8	ARM Coretex™-A8	ARM Coretex™-A8	ARM Coretex™-A8
32ビット 固定小数点 DSP	TMS320C64X+™	TMS320C64X+™	-	-
マルチメディア サブシステム	IVA2.2	IVA2.2	-	-
2D/3Dグラフィックス アクセラレータ	POWERVR SGX™	-	POWERVR SGX™	-
イメージ・シグナル・プロセッシング（ISP）	共通			
ディスプレイ サブシステム	共通			
ペリフェラル	共通			

各製品ともコアを共通化しており，開発資産の再利用が可能である．また共通ペリフェラルをピン互換で配置することで，最終製品の基板を変更せずに仕様の異なる製品を開発することが可能である．

Linux 評価モジュールには，Linux カーネル 2.6.22 ボードサポートパッケージや，ソフトウェアデベロップメントキット（SDK），ペリフェラルドライバブートローダ，ルートファイルシステム，評価ツールが含まれる．ドータカード拡張コネクタが実装されているため開発ボード（EVM）を使用して最終製品のプロトタイプを開発することも可能である．

図 2･41 に示す OMAP 3530 を例にプロセッサを説明する．

OMAP3530 は浮動小数点処理 SIMD ユニットをもつ 32 ビット RISC プロセッサの MPU サブシステムと，32 ビット固定小数点 DSP サブシステム，マルチメディアサブシステム（IVA2.2），2D/3D グラフィックスアクセラレータ，カメライメージシグナルプロセッシング（ISP），ディスプレイサブシステム，各種インタフェースで構成される．

RISC プロセッサは逐次型デュアル発行スーパースカラ方式で, 命令を順に読み込んで複数のパイプラインで並列実行が可能であり，同じ周波数の旧世代プロセッサと比較した場合，同一周波数で 2 倍程度の処理能力がある．

DSP サブシステムは DaVinci でも採用されている C64x+コアを搭載しており各種ディジタル信号処理を実行する．マルチメディアサブシステム（IVA 2.2）と協調して H.264 720p（1280×720 ピクセル）を毎秒 30 フレーム，デコードすることが可能である．

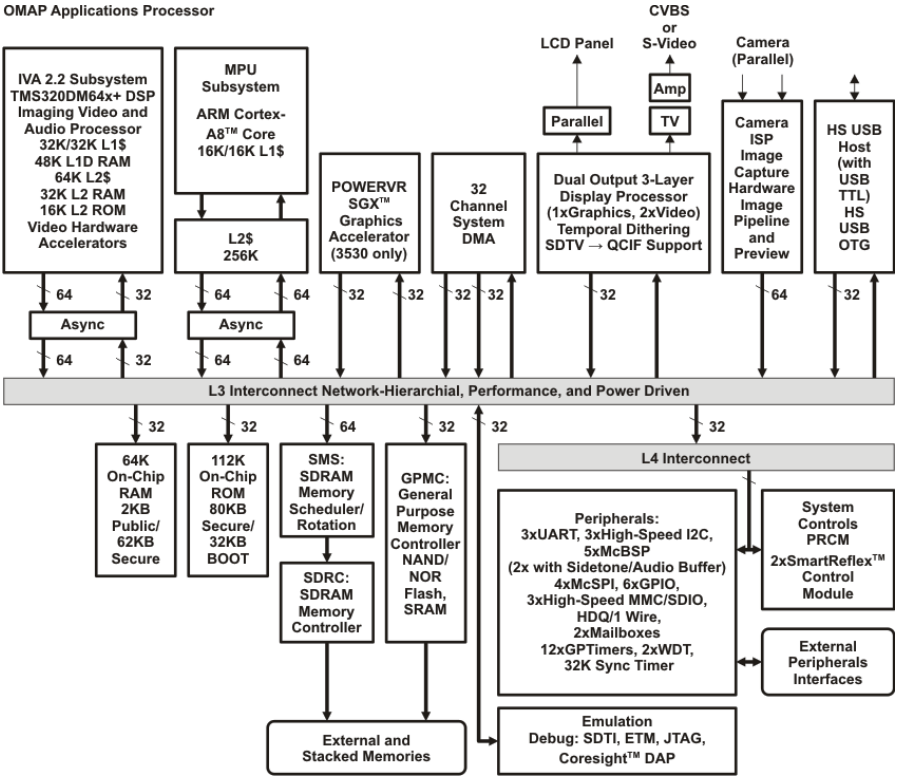


図 2・41　OMAP 3530

グラフィックスアクセラレータは組込み機器用の 3D グラフィックス API（OpenGL ES2.0／1.1）と，2 次元ベクトルグラフィックス描画用 API（OpenVG 1.0）に対応している．32×32 ピクセルを一つのタイルとして演算を行うタイルアーキテクチャと，ピクセル演算（画素処理）とバーテックス演算（頂点処理）をマルチスレッド化し，一つのユニットで演算実行するアーキテクチャを採用している．

ISP は CCD/CMOS センサからの RAW や YCbCr などのディジタル信号やビデオデコーダからのカメラ信号を取り込み，画像の逐次処理が可能である．ヒストグラム出力やリサイズも行う．

ディスプレイサブシステムは 4 面（ビデオ 2 面，ビットマップ 2 面）の画像重畳が可能であり，24 ビット RGB のパラレルディジタル出力に対応し最大二つの液晶ディスプレイパネルに接続可能である．また NTSC 形式／PAL 形式のテレビ出力が可能なビデオエンコーダを備え S-Video にも対応している．

ペリフェラルは，外部メモリインタフェース，USB2.0HS OTG，マルチチャンネル，シリアル，ポートインタフェース（McSPI），マルチチャネルバッファードシリアルポート（McBSP），UART，I2C ほか最大 188 ピンの汎用 I/O を備えている．

(5) TI の開発環境

図 2･42 に TI の開発環境の全体を図示する．

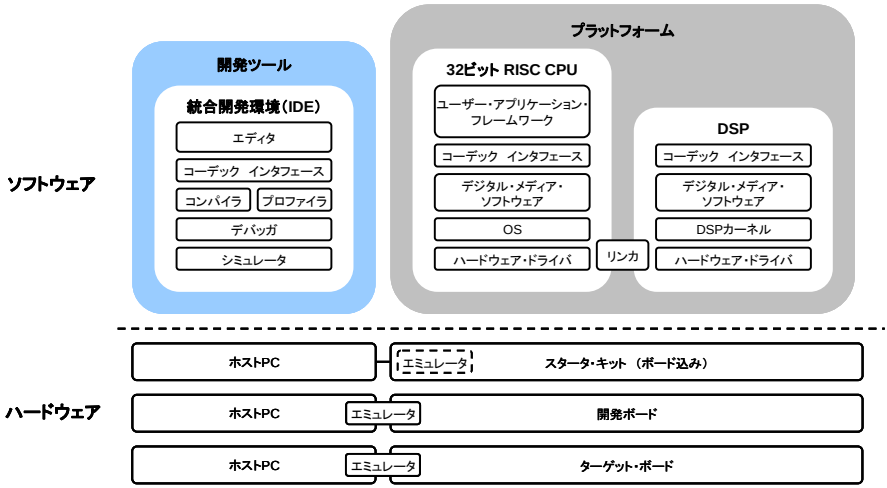


図 2・42　開発環境　全体図

統合開発環境（IDE）や，無償評価用ツール，リアルタイムカーネル，エミュレータ／アナライザ，EVM などを開発ツールとして提供している．IDE はエディタ，コンパイラ，デバッガなどで構成される．開発用ソフトウェアは CPU 用 OS や DSP 用のカーネルのほか，ペリフェラルドライバ，チップサポートライブラリ（CSL），ネットワーク開発キット（NDK）が提供される．また，ディジタルメディアアプリケーションの評価用としてディジタルメディアソフトウェア（DMS）が用意される．DSP の評価に必要な開発ツールとボードを含んだ DSP スタータキット（DSK）や，ディジタルビデオアプリケーションの開発が可能なディジタルビデオ評価モジュール（DVEVM）など，目的に応じてハードウェアとソフトウェアを組み合わせた製品も提供もしている．

更に，量産に向けた特定アプリケーション用途のライブラリの拡充にも注力しており，画像解析や，音声特殊効果，特殊演算フィルタなどを開発，提供している．

TI は開発環境の標準化，オープン化の推進と合わせて，世界各国で開発パートナー会社を組織化し，開発を支援している．プロセッサの開発においても開発資産の再利用を想定し，設計仕様の継続性を重視することで，TI のプラットフォームで開発に注力できる環境を提供している．これらを背景に開発パートナー会社からは，コーデックをはじめとするディジタルメディアソフトウェアや，特定アプリケーション向け開発キット／ライブラリ，ネットワークやメディア対応のミドルウェアなど専門性の高い優れた製品が提供され，各種オペレーティングシステムもサポートされている．また TI のプラットフォームやコアに精通したパートナー企業による受託開発サービスや開発ツール，レファレンスデザインなども提供されており，これらを活用することで開発費用を抑えながら，短期間で最終製品を開発することが可能となる．

(6) おわりに

2011 年 7 月 24 日に地上テレビ放送が完全ディジタル化という大きな節目を迎える．今後の社会における情報処理のディジタル化は自然な流れであり，処理能力の継続的な向上要求は避けて通れない．一方で携帯電話に例示されるように，新興国を含めた市場の拡大にともない低価格要求を招いている．更に情報機器のコモディティ化が進み，差異化の難しさから最終製品の早期市場投入が求められ，製品開発におけるプラットフォーム化が進んでいる．

TI のロードマップは社会の要求に同調しており，プロセッサのマルチコア化と動作周波数の向上，各種ハードウェアアクセラレータの拡充，ファインプロセスの採用，及び電力制御・管理技術の改良による低消費電力化を進め，各種ソフトウェア群を含む開発キットのリリースと統合開発環境の改良などにより，プラットフォームを更に発展させていく予定である．

TI では製品構成上，アナログ製品群とディジタル製品群がシェアを等分している．TI では，自然界の情報の多くがアナログであることから，アナログ信号をディジタル信号に変換し，ディジタル処理の後，再びアナログ信号へ変換し出力する，という自然な情報処理を重視している．このシグナルチェーンの中にディジタル信号処理を位置付けることで，更なる付加価値を提案していくことが可能となる．

■参考文献

1) 生駒伸一郎, “DSP 入門講座,” 電波新聞社.
2) 服部基保, “マルチメディア処理向け DSP TMS320C6000 活用ハンドブック,” CQ 出版社.
3) 電子ジャーナル “2003 システム LSI 技術大全”
4) テキサス・インスツルメンツ, “TMS320DM6446 Digital Media System-on-Chip（SPRS283F）”
5) テキサス・インスツルメンツ, “OMAP3530/25 Applications Processor（SPRS507C）”
6) テキサス・インスツルメンツ, “OMAP35x Applications Processor Introduction（SPRUFF1B）”
7) テキサス・インスツルメンツ, “DaVinci™ テクノロジー 概要（JAJB019）”
8) テキサス・インスツルメンツ, “Code Composer Studio 開発ツール v3.3 入門マニュアル”
9) テキサス・インスツルメンツ, “TI のスタータ・キット（JAJW007）”

2-6-6 組込み用途向けマルチコアプラットフォーム

（執筆者 須賀敦浩・山内宏真）［2009 年 6 月 受領］

高性能と低電力の両立が強く要求される組込みシステムでは，動作周波数を抑えながら演算性能の向上が可能な並列処理は有効な解決方法の一つである．従来，VLIWなどの命令レベル並列性やSIMDなどのデータレベル並列性が利用されてきたが，4～8 程度の並列度が限界であり，更なる性能向上のために，マルチコアの各コアに処理を割り当てるタスクレベル並列性が用いられるようになってきた．FR 1000 は，VLIW型アーキテクチャによる 8 ウェイ命令レベル並列性と 4 ウェイのSIMD型命令を実装したFR-V[*1]（**図 2･43**）ファミリー“FR 550 プロセッサコア”を四つ搭載したマルチコアプロセッサである [1].

図 2･44 に FR 1000 のブロック図を示す.

[*1] FR-V：富士通株式会社，及び株式会社富士通研究所が開発している組込み用途向けマルチメディアプロセッサ.

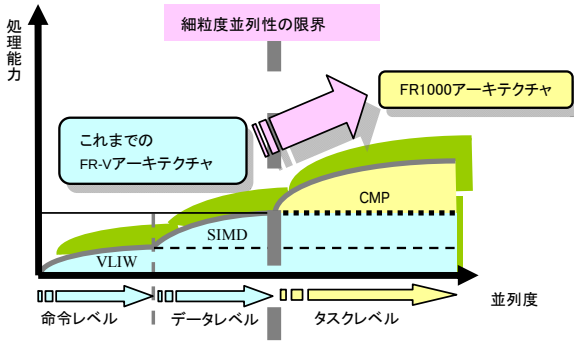


図 2・43　FR-V プロセッサの変遷

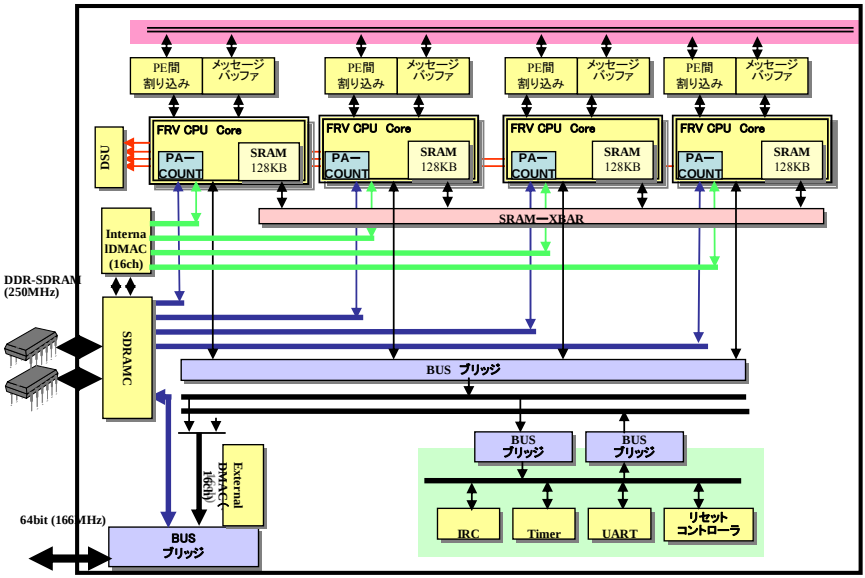


図 2・44　FR 1000 ブロック図

マルチコアプロセッサにおいて，プロセッサの演算能力の最大限に引き出して高いシステム性能を実現するために重要となるのが，データ転送とプログラミングモデルである．特に，HDTV システム，高速・高精細のプリンタシステムやグラフィックシステムのような大きな画像データを扱うシステムでは，高いデータ転送性能を実現が必要となる．そのため，FR 1000 では下記のような構造を備えている [2]．

第 1 は，各 CPU コアに対するローカルストレージとして 128 K バイト SRAM の搭載である．これにより，主記憶へのアクセスを減らし，高速なメモリアクセスを実現している．

第 2 は，四つのプロセッサコア，二つの DDR-SDRAM とメモリ IF，及び一つのシステム

バスと IF 間の接続のクロスバ化である．ローカルストレージ同士は専用のクロスバで接続し，各 CPU コアは，ほかの CPU コアのローカルストレージに直接アクセスすることが可能となっている．

第 3 は，CPU コア間のコマンド転送専用通信制御機構の内蔵である．各 CPU コアには，CPU コア間通信専用のメッセージバッファを用意し，4 バイトのメッセージデータを 9 サイクル以内にほかの CPU コアにデータ転送とは独立に送信できる．

第 4 は，DMA コントローラの内部用と外部用への分離である．内部用 DMA コントローラは CPU コア間，CPU コアと主記憶の間，主記憶と主記憶の間の転送を司る．一方，外部用 DMA コントローラは，主記憶とシステムバスの間の転送を担う．それぞれ 16 チャネルを同時に制御可能である．こうしたバスアーキテクチャにより，ローカルストレージ間，主記憶上の領域同士，主記憶と外部デバイスとの間での同時データ転送を実現している．

一方，プログラミングモデルとしては，動的負荷分散スケジューラを用いた非同期遠隔手続き呼び出し（ARPC：Asynchronous Remote Procedure Call）による ARPC プログラミングモデルが提案されている．ARPC プログラミングモデルは，シングルコア向けに開発したソフトウェア資産を最大限再利用し，搭載するコアの数や種類の変更に対するソフトウェアの修正なしに性能向上を目指したプログラミングモデルである．

組込みソフトでは，独立に動作する多くの機能を関数単位で実装していることが多いため，特殊な記述をしなくても高い並列性を備えている．ARPC プログラミングモデルは，分割した機能単位の処理を非同期遠隔手続き呼び出しによって複数のコアで並列処理するモデルであるが，関数を並列処理の単位とすることで，シングルコア用のプログラムの変更を極力抑え，ソフトウェアの再利用性を高めている．

また，動的負荷分散スケジューラは，実行時に各コアの処理状況に応じて負荷が均等になるようにコアを決定し機能の割り当てる．これにより，プログラムにおいて割当先のコアの記述が不要となり，コアの数・種類の異なる様々なマルチコアプロセッサに対するソフトウェアのスケーラビリティを実現している．

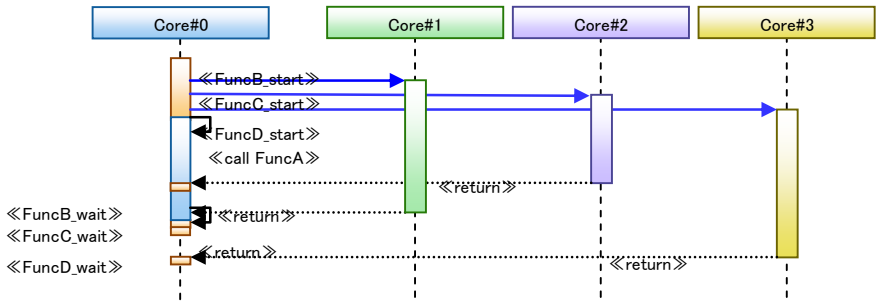


図 2・45　ARPC プログラミングモデルにおけるシーケンス

動的負荷分散スケジューラの実現方法としては，ソフトウェアとハードウェアの両方の方法がある．ソフトウェアで実現する場合，プログラムの機能を処理するコアとスケジューラ

のコアを共通化することで回路量を削減できるが，コア数の増大などによりスケジューリングの処理量が多くなると，コアに占めるスケジューリングの処理時間やコアの性能が問題となる．これに対し，ハードウェアでの実現により，スケジューリングの処理量の増大に対応することが可能となり，コアにおけるスケジューリング処理の負荷の問題も解決する．**図 2･45** に ARPC プログラミングモデルのシーケンスを示す.

■参考文献

1) 須賀敦浩, “スーパーコンの技術を民生機器向けプロセサに適用　マルチコア向け性能チューニングの勘所,” 日経エレクトロニクス, 2005 年 9 月 12 日号.
2) 須賀敦浩, “スケーラビリティを実現する組み込み用途向けマルチコアプラットフォーム,” マイクロプロセッサフォーラムジャパン, 2008.