2004－ARC－157　(3)
2004－HPC－ 97　(3)
2004／3／1

マルチメディアネットワーキング向きデータ駆動プロセッサのLSI試作

伊藤伸也[†]　野本祥平[†]
冨安洋史[††]　西川博昭[††]

著者らは，データ駆動・制御駆動スレッドを命令レベルで同時・多重処理し，データ駆動の公平な多重処理能力を維持しつつ逐次部を高効率化したマルチメディアネットワーキング向きプロセッサ CUE-v2 の LSI 試作を行っている．CUE-v2 は，データ駆動プロセッサ中の発火制御機構がリザベーションステーションに似ていることに着目し，共通のパイプラインでデータ駆動と out-of-order スーパスカラの双方の実行が可能なよう設計されている．2 種類のスレッドの同時・多重処理を可能とするため，特に，命令フェッチ機構および発火制御機構に特徴がある．本論文は，CUE-v2 のハードウェア設計およびその規模について述べている．

The LSI Implementation of A Multimedia Networking Oriented Data-Driven Processor

Shinya ITO ,[†] Shouhei NOMOTO ,[†] Hiroshi TOMIYASU [††]
and Hiroaki NISHIKAWA[††]

The authors have been developing the CUE-v2 LSI chip performing both as data-driven and as superscalar processor with a common pipeline. The CUE-v2 can simultaneously process data-driven and control-driven threads at instruction level. With this scheme, the CUE-v2 can inherit the advantage of data-driven processor, which is the fair multiprocessing, and achieve effective sequential processing. Although the basic design of the CUE-v2 has many similarities to that of superscalar, instruction fetch unit and the scheme of dynamic out-of-order scheduling are differ considerably due to the simultaneous execution of 2 kinds of threads. This paper describes the design and the hardware amount of the CUE-v2.

1. はじめに

近年の通信伝送路技術の発展により，ネットワークの帯域は著しく増加している．これに伴い，ヘッダ領域のみが処理対象である従来からのパケット転送処理だけではなく，メディアトランスコーディング[1] のようにペイロード領域も処理対象とする高度な処理をルータで行うことが検討されている[2]．このようなマルチメディアネットワーキング処理を実現するには，メディアストリームを同時に複数扱いつつその実時間性を維持した上で，パケット転送することが求められる．

著者らは，プロトコル・メディア処理に内在する全ての並列性を自然に活用可能，かつ，その実時間多重処理をアーキテクチャ水準で実現可能なプロセッサとして，データ駆動プロセッサを研究してきた．これまでに，データ駆動プロセッサの実機 CUE-v1[3] を用いて，OC-3 ATM における CORBA プロトコル処理の off-loading および VGA サイズの実時間動画像圧縮処理を実現した[4][5]．その結果，パイプラインを過負荷としなければ，実行時のソフトウエアスケジューリングなしにターンアラウンドタイムを一定に維持可能であることが示された．すなわち，実際のマルチメディアネットワーキング処理時にも各プロセスの実行時間が予測可能かつプロセス増加によるオーバヘッドが無く，実時間多重処理をアーキテクチャ水準で実現可能であることを実証した[4][5]．

一方，スループットの観点からデータ駆動プロセッサを見たとき，計算の局所性を無視しているため逐次部の実行が非効率であることが以前から指摘されている[6][7]．実際に，これまでの研究から，プロトコル処理ではコネクションの管理や送出ポートの共有時の管理等，メディア処理ではストリーム生成のためのパラメータの直列化等において，逐次処理が不可避な箇所が存在し，これらがボトルネックになっている．そこで，データ駆動プロセッサの有する，実時間多重処理能力を維持しつつ逐次処理の高効率化を可能とする実行方式を提案した[8]．本方式では，データ駆動命令を優先的に発行した上で，データ駆動・制御駆動の両スレッドを同一パイプラインで命令単位に多重処理する．本論文では，現在 LSI 試作中のマルチメディアネットワーキング向きデータ駆動プロセッサ CUE-v2 の実際の設計およびそのハードウェア規模について述べる．

† 筑波大学 システム情報工学研究科
Doctoral Program in Systems and Information Engineering, University of Tsukuba
†† 筑波大学 電子・情報工学系
The Institute of Information Sciences and Electronics, University of Tsukuba

2. CUE-v2 の動作方式と構成

2.1 データ駆動・制御駆動スレッドの同時・多重処理

CUE-v2 は，異なる性質を持つ 2 種類のスレッドを同一パイプライン上で命令レベルで同時・多重処理することにより，以下の 2 点を両立可能なプロセッサアーキテクチャの確立を目指している．1) データ駆動スレッドの実行により，データフローグラフから示される並列性を最大限に活用し，並列部を高効率に実行する．さらに，データ駆動プロセッサの特長である，公平な多重処理を活用した実時間多重処理を実現する．2) 制御駆動スレッドの実行により，計算の局所性を活用し，逐次部のパイプライン処理を可能とし，その実行時間を通常のフォンノイマン型プロセッサと同等とする．

CUE-v2 は，基本的にデータ駆動スレッドによる実行を行うが，アプリケーション中の逐次処理が避けられない箇所を制御駆動スレッドにより実行する．本方式のプログラミングは，基本的にデータフローグラフの作成によって行う．そして，性能を制限する逐次処理部を制御駆動命令に置き換える．本方式の開発支援環境および最適プログラミング手法については現在検討中であり，現在のところアセンブラによるプログラミングのみのサポートである．

本論文では，データ駆動・制御駆動スレッドの各スレッドを以下のように定義するものとする．

データ駆動スレッド

同一のカラーを有するトークンの実行シーケンス．命令の発行は，従来のデータ駆動プロセッサと同様にデータ依存関係に基づく．

制御駆動スレッド

プログラムカウンタ (PC) に基づき連続的に発行された命令の実行シーケンス．複数の制御駆動スレッドの実行は起動順に non-preemtive に行うものとする．

CUE-v2 では，これら 2 種類のスレッド間で相互に干渉することなく命令単位で多重・同時処理を可能とする命令フェッチポリシーを採用している．本ポリシーでは，基本的にデータ駆動命令を制御駆動命令に対し優先的にフェッチ・発行する．すなわち，CUE-v2 では，データ駆動スレッドの実行時に生じる空きパイプライン資源を制御駆動スレッドに割り当てる．詳細は 2.3.1 に示す．また，異なる種類のスレッドの起動およびデータの受渡しは以下のとおりである．まず，データ駆動→制御駆動時は，任意の PC 値で PC を有効とする命令を実行し，その後，データ駆動パケット中のデータ部をレジスタに格納する命令を用いデータを渡す．受け渡すデータが多いときは，メモリを介したポインタ渡しとする．また，制御駆動スレッドの最後の命令は，必ず PC を無効とする命令とし，スレッドの終了を明示する．制御駆動→データ駆動時は，データ駆動パケットを生成する命令を実行することにより，起動およびデータ受渡しを行う．受け渡すデータの数が多いときは，ポインタ渡しとする．なお，データ駆動スレッドは，その原理上，スレッドの終了を明示する命令は必要ない．さらに，上記の PC 有効化命令およびデータ駆動パケット生成命令は，プログラム中の任意の位置で任意の回数，使用できる．

CUE-v2 と従来のデータ駆動・制御駆動のハイブリッドプロセッサとの動作方式上の大きな違いは以下のとおりである．従来のハイブリッドプロセッサでは，scheduling quanta[6] や強連結ブロック[7] と呼ばれる制御駆動に基づく実行部分を，排他的に実行する．このため，データ駆動プロセッサの有する公平な多重処理という特長を阻害する可能性がある．さらに，従来のハイブリッドプロセッサでは，同一パイプライン上でデータ駆動・制御駆動を混在させる場合，実行中のデータ駆動スレッドの中断・再開を行う必要があるが，データ駆動には順序と言う概念が無いため，制御駆動スレッドの中断・再開と比較して実装が複雑となる．実際に，EM-4[7] では，混在させず，循環パイプラインと強連結パイプラインは別個に設けられている．

2.2 命令セットアーキテクチャ

CUE-v2 は，データ駆動命令 123 個，制御駆動命令 105 個の計 228 命令を持つ．その種類は，算術・論理，シフト，LZD(Leading Zero Detection)，メモリアクセス (カラーによるアドレッシングも可)，カラー操作，分岐，スレッド起動・終了関連，である．CUE-v2 は，32 bit のオペランドに対して演算を行う．また，乗算を除く算術，論理，シフト命令に関しては，8bitx4 および 16bitx2 の SIMD 演算もサポートする．

CUE-v2 の命令形式は，32bit 固定長であり，図 1 に整数演算命令 (INT) の例を示す．データ駆動命令と制御駆動命令の間の主な相違は以下の通りである．1) データ駆動命令は全ての命令が分岐命令に相当するため，分岐先 (dest) を持ち，さらに，その左右 (lr) を区別する．また，条件分岐時のオーバヘッド削減のため，コンディションコード (cc) 付き命令としている．2) 制御駆動命令は，レジスタ識別子 (rd，rs0，rs1) を持つ．

なお，データ駆動命令と制御駆動命令の区別に関しては，命令形式中に明示するビットを設けるのではなく，命令フェッチ機構において，循環パスを介したデータ駆動パケットの到着によりフェッチされるのか PC 値を用いてフェッチされるのかにより識別している．

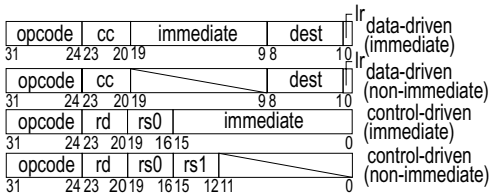

図 1 CUE-v2 の命令形式

2.3 マイクロアーキテクチャ

CUE-v2 は，データ駆動プロセッサ中の発火制御機構がリザベーションステーションに似ていることに着目し，共通のパイプラインでデータ駆動と out-of-order スーパスカラの双方の実行が可能なよう設計されている．ただし，CUE-v2 はアーキテクチャ水準での実時間多重処理を目的としている為，制御駆動スレッドの投機実行はミスペナルティが大きいので行わない．

CUE-v2 のパイプライン構成を図 2 に，全体構成を図 3 に示す．CUE-v2 は，従来のフォンノイマンプロセッサの見地から見れば，2 命令同時発行の out-of-order スーパスカラに 2 種類のスレッドを管理する機構 (DiCount，CTQ(Control-driven Threads Queue)) およびデータ駆動スレッド用の循環パスを設けた構成といえる．逆に，従来のデータ駆動プロセッサの見地から見れば，フロン

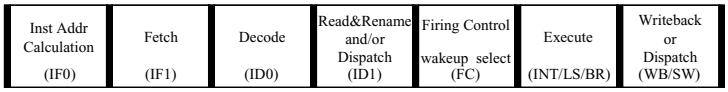

図 2 CUE-v2 のパイプライン構成

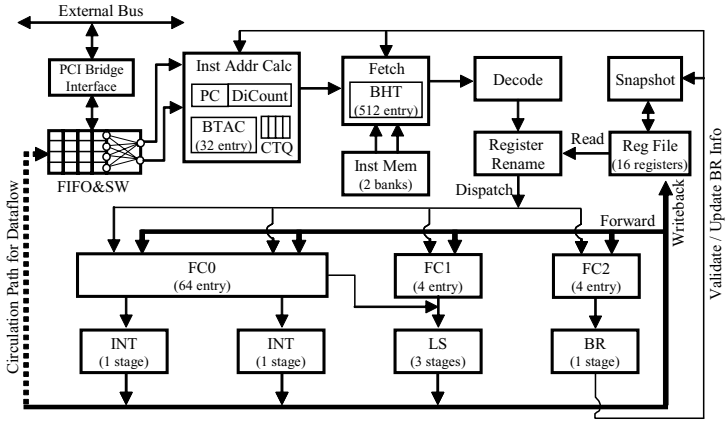

図 3 CUE-v2 の全体構成

トエンド部にプログラムカウンタ (PC) に基づく命令発行およびレジスタによるデータ受渡しを追加した構成といえる．データ駆動・制御駆動の 2 種類のスレッドの実行にできるだけ共通のハードウェアを用いるため，片一方でのみ使用するモジュールは最小限に留めている．たとえば，レジスタの本数を 16 本と制限し，比較的小さい分岐予測器しか設けていない．CUE-v2 のパイプラインは，データ駆動・制御駆動の双方において，INT 命令で 7 段，LS(Load/Store) 命令で 9 段，BR(BRanch) 命令 (制御駆動のみ) で 6 段である．以下に，各パイプラインステージの機能を解説する．

IF0 は，データ駆動・制御駆動スレッドのどちらを実行するかを選択し，選択されたスレッドの命令アドレスを IF1 に転送する．IF1 は，IF0 で計算されたアドレスに基づいて，命令メモリ (Inst Mem) から命令をフェッチする．本命令メモリは，2 バンク構成をとり，偶数番地，奇数番地の命令を同時にフェッチできる．ID0 は，IF1 でフェッチされた命令をデコードするとともに，FC の待ち合わせメモリに当該命令を格納するための領域を確保し，そのアドレス (MMA) を確保する．なお，MMA はデータ駆動命令の場合は FC からの出力時に，制御駆動命令の場合は WB への入力時に，開放する．ID1 は，ID0 で確保した MMA に応じて，入力された命令を適切な行き先に dispatch する．さらに，制御駆動命令の場合は，レジスタ読み込みおよび MMA に基づくレジスタリネーミングを行う．すなわち，リネーミングのタグとして MMA を用いる．FC は，データ駆動・制御駆動の両スレッドの out-of-order スケジューリングを行い，演算に必要なソースオペランドを待ち合わせる．さらに，制御駆動スレッドに関しては，CUE-v2 では投機実行を行わないため，FC にて，先行する分岐命令の分岐方向が確定するのを待ち合わせる．INT，LS は，入力された命令の実行を行う．データ駆動命令の場合は，次の命令の飛び先を計算した後に SW に転送し，命令を循環させる．また，SW に同時に到着するデータ駆動命令数は，最高 4(INTx2，LSx1，PCIx1) である一方で，SW-IF 間のデータパスは 2 本であるため，スイッチの前に FIFO を設けている．本 FIFO はパス毎に 4 エントリを有する．制御駆動命令の場合は，WB に転送しレジスタに値を書き戻すとともに，演算結果を FC にフォワーディングする．BR は制御駆動スレッド時のみ利用され，制御駆動の分岐命令を実行し，その分岐結果を分岐予測に関わる機構および分岐方向の確定を待ち合わせる FC に送る．BTAC(Branch Target Address Cache) および BHT(Branch History Table) はそれぞれ分岐先アドレス予測器と分岐方向予測器である．Snapshot は，分岐予測失敗時のレジスタファイル (Reg File) のリカバリに用いる CUE-v2 独自の機構である．また，分岐予測失敗時には，フロントエンド部の各ステージおよび待ち合わせメモリに対してリカバリ要求を出し，in-flight 状態の制御駆動命令をパイプライン中から除去する．CUE-v2 の設計において特に特徴的である，命令フェッチ，発火制御，メモリアクセスについて以下に詳解する．

2.3.1 命令フェッチ

CUE-v2 では，データ駆動・制御駆動スレッド間の資源分配が重要である．マルチスレッド実行時の資源分配は，主に，命令フェッチポリシーに依存する．たとえば，文献[9] では，SMT において ICOUNT と呼ばれる機構を設けることにより，多くのスロットを埋めることが可能なことを示している．CUE-v2 では，同時もしくは多重に実行するスレッドがデータ駆動スレッドと制御駆動スレッドの 2 種類であり，その性質が大きく異なるため，スレッド種間の命令フェッチポリシーが特に重要となる．CUE-v2 ではデータ駆動の特長を維持するため，基本的にデータ駆動スレッドの命令を優先的にフェッチし，フェッチ可能なデータ駆動スレッドが存在しない場合に制御駆動スレッドをフェッチするもの，とした．た

だし，データ駆動スレッドの並列性が高くなりスロット数以上となる場合，制御駆動スレッドを実行する計算資源がなくなるため，制御駆動スレッドの実行時間が著しく増大する可能性がある．ゆえに，一定の割合 (n 回に 1 回) で制御駆動スレッドをフェッチすることを保証するものとする．CUE-v2 では，このフェッチポリシーは，連続してデータ駆動命令をフェッチする回数をカウントする DiCount により管理される．DiCount は 4 bit の飽和インクリメンタである．また，制御駆動命令を発行する毎に DiCount 値は 0 にリセットされる．DiCount の飽和値，すなわち，制御駆動命令のフェッチ保証頻度 n は，デフォルトでは INT パイプライン長である 7 としている．さらに，飽和値は，専用命令によりプログラム中で $0 \leq n \leq 14$ の範囲で変更可能である．なお，$n = 0$ のときは，データ駆動スレッドが常に優先となる．また，$n = 15$ のときは，制御駆動スレッドの排他的実行を行う実験用モードとしている．

CUE-v2 では，各制御駆動スレッドは non-preemptive に実行されるため，先行するスレッドの終了を CTQ で待つ．CTQ は 32 エントリの FIFO で構成されている．また，CUE-v2 は，分岐予測器として，BTAC および BHT を有する．CUE-v2 は，投機実行を行わないため，90% 程度の分岐方向予測の精度を目標としている．そこで，分岐予測精度に関する予備的評価を，SimpleScalar Tool Set ver. 3.0[10] を用いて行った．同様の構成のスーパスカラにおける SPECint95 中の 8 種のベンチマークで評価した．その結果，BTAC32 エントリ，bimodal 方式の BHT512 エントリの構成で，相乗平均値で，分岐先アドレス予測精度 65.25%，分岐方向予測 90.58%となった．この結果を基に，各分岐予測器のエントリ数を決定した．

図 4 は命令フェッチ機構の構成を示し，以下に，IF におけるスレッド選択の実装を説明する．データ駆動・制御駆動スレッドのどちらの命令をフェッチするかに関しては，上述の DiCount からの信号 sel_ct の値により決定する．また，PC が有効である限り (pc_active=1)，新たな制御駆動スレッドが起動されることはない．PC が有効であるとき，PC 値は PC+1，BTAC の予測アドレス，BHT の予測に基づくアドレスが候補となる．これらのどれを選択するかに関しては，分岐状態に応じた選択を指定する Priority Encoder を持つ．新しく起動される制御駆動スレッドの命令フェッチに関しては，CTQ が空かどうか (ctq_empty) に応じて選択する．CTQ が空ではなかった場合は，CTQ 内の制御駆動スレッドの起動命令をフェッチするものとし，新規に起動要求の出された制御駆動スレッドは CTQ に格納される．CTQ が空の場合は，新規の起動命令をフェッチする．

2.3.2 発火制御

CUE-v2 は，発火制御する演算の種類に応じて，FC0，FC1，FC2 の 3 つの発火制御機構を持つ．FC0 は，64 エントリの待ち合わせメモリを有し，データ駆動・制御駆動の双方の整数命令およびデータ駆動のロード・ストア命令の発火制御を out-of-order で行う．FC1 および FC2 は，4 エントリの待ち合わせメモリを有し，それぞれ，制御駆動のロード・ストア命令および制御駆動の分岐命令の発火制御を in-order に行う．FC1 および FC2 において，発火制御を in-order に行うことにより，ロード・ストアおよび分岐命令の実行順序を保存し，プログラムのセマンティックスと異なる動作を防いでいる．

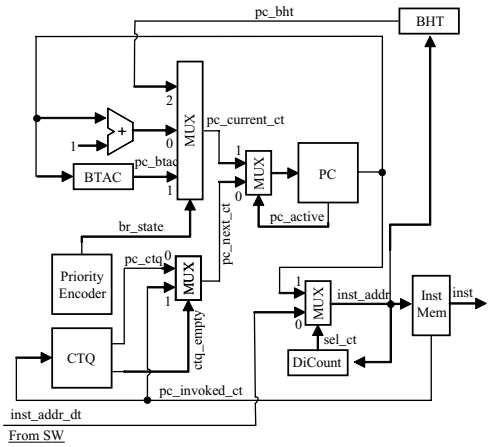

図 4 CUE-v2 の命令フェッチ機構の構成

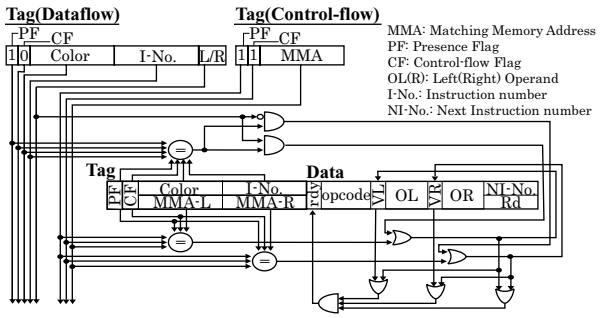

図 5 CUE-v2 の待ち合わせメモリの基本構成 (説明の簡略化の為，1 本のフォワーディングパスとしている．)

CUE-v2 の FC は，通常の out-of-order のスーパスカラプロセッサと同様に，命令の実行に必要なオペランドが揃ったことを検出する Wakeup 回路と発火可能な命令を選択する Select 回路から構成される[11]．ただし，CUE-v2 では，データ駆動・制御駆動の双方において共通の発火制御機構を利用する都合上，以下の 2 点において通常のリザベーションステーションと異なる構成をとる．(1) 演算器からのフォワーディングパスのデータと待ち合わせ記憶内に蓄えられたデータの間の Wakeup のみならず，ID1 ステージから到着するデータ駆動パケットと待ち合わせ記憶内のデータの間の Wakeup も検出する．(2)Select 回路において，待ち合わせ記憶内での検索開始位置を常に移動させることにより，データ駆動命令が長時間留まることを防いでいる．out-of-order スーパスカラプロセッサにおいて待ち合わせでのセレクションポリシーが性能に大きな影響を与えないことは，文献[12] 等で既に指摘されているが，CUE-v2 のように異なる種類のスレッドが同時に複数動作される環境においても，同様の結果となるかどうかは不明である．また，CUE-v2 では，実時間多重処理の都合上，データ駆動パケットが長時間待ち合わせに滞在することを防ぎたい．ゆえに，前回セレクトされた待ち合わせメモリのアドレス (MMA)+1 を検索開始位置とするセレクションポリシーを付加した Select 回路とした．

図 5 に，待ち合わせメモリの構成を示す．待ち合わせメモリは，マルチポートの連想記憶より構成され，Tag 部と Data 部からなる．データ駆動命令の Wakeup は，ID1 から入力される，カラー (Color) および命令番号 (I-No) からなるタグの一致検索により行われる．制御駆動命令の Wakeup は，演算器からのフォワーディングパスから入力される，オペランドを供給する先行命令の MMA をタグとして一致検索することにより行われる．Wakeup により発火可能となったエントリは，そのことを示す信号 rdy を 1 にセットするとともに，Select 回路に対し，それを通知する．FC0 の Select 回路では，64 本の rdy 信号からなるビットベクトル (rdy_vector) を受け取り，その中からセットされているものを 2 つ選択する．2 命令の選択を行う方法としては，64to1 のセレクタを直列に接続する方法[11] があるが，この方法では速度の点で問題が生じた．そこで，CUE-v2 では，MMA の偶奇により分離した 2 つの 32to1 セレクタを採用している．また，上記 2）のセレクションポリシーを実現するため，以下のように実装した．前回セレクトされた命令の MMA を MMA_p とする．MMA_p+1 以降からセレクトする回路と先頭から MMA_p までセレクトする回路を用意し，これら 2 つのセレクト結果のうち前者を優先的にセレクトする回路を用意した．MMA_p は，Select 回路内に，0 から MMA_p まで全て 0，MMA_p+1 以降は全て 1 の，ビットベクトルとして保存している．そして，(rdy_vector & MMA_p_vector) と (rdy_vector & ~(MMA_p_vector)) を 2 つのセレクタに入力している．この実装法のため，Select 回路が大きくなる．最後のセレクタの分遅延が大きくなるが，これは，それほど大きな値ではない．

2.3.3 メモリアクセス

データ駆動プログラムでは多数のプロセスが命令単位で実行されるため，メモリアクセスの局所性を生かすことが難しい．そこで，CUE-v2 では，キャッシュを設けず内蔵メモリをスクラッチメモリとして用い，ソフトウエアで明示的に外部メモリを読み書きするようにした．このため，ロードストアユニットはスループットを重視し，かつ外部メモリ読み書き中でもスクラッチメモリをアクセスできるようにした．

CUE-v2 は外部メモリとして SDRAM を採用し，burst 読み書きを用いた内蔵メモリとの転送のみをサポートする．外部メモリをアクセスする命令はレイテンシが大きくなるため，内蔵メモリのポートが長時間占有されないようにしなければならない．このため内蔵メモリを 2 バンク構成とし，少なくとも片方のバンクが使用可能になるようにした．

このような機構を実現するため，LS のパイプラインを 3 段で構成した．まず初段でアドレス計算とアドレスの衝突検出を行う．ここで衝突を検出することによって，先に発行された外部メモリアクセス命令と，後続の内蔵メモリアクセス命令が競合しないようにしている．二段目ではメモリアクセス要求を行う．内蔵メモリアクセス命令の場合はそのままアクセスし，外部メモリアクセス命令の場合は専用のユニットへ制御を移す．この外部メモリアクセスユニットは外部メモリをアクセスすると共に，内蔵メモリとの転送を独立した 1 word 毎のアクセスに分解する．このため，外部メモリアクセス命令が長時間内蔵メモリのポートを占有することが無いようになっている．設計とライブラリ上の制限のため，今回は内蔵メモリとして，同期メモリを同相クロックで使用した．このため，内蔵メモリのアクセス結果は三段目で得られる．

3. CUE-v2 の LSI 試作

CUE-v2 の性能，ハードウェアコストおよび設計難易度を評価するために，実際に LSI 試作を行っている．最終的には，今回試作中のコアを複数搭載するチップマルチプロセッサとして実現することを想定している．しかし，今回は，研究用のプロトタイプであり，また，大学での LSI 試作であるため，設計人員・経験・予算の兼ね合いから，1 つのコアのみの設計・試作とした．本論文では，この試作中の LSI を CUE-v2 チップと呼ぶ．

本稿執筆時における CUE-v2 チップの設計の進捗状況は，レイアウトを開始する直前の段階である．この段階での RTL シミュレーション，論理合成，STA(Static Timing Analysis)，ゲートレベルシミュレーションを既に行っている．テストパターンとして，各機能確認用のものの他に，アプリケーションレベルの挙動を確認するために，IP-send，ADPCM エンコーディング，CRC，行列積 (50x50)，バブルソート等をアセンブラ記述し，これらを基に生成した．これらのテストパターンは，1)2 種類のスレッドの同時・多重実行，2) 制御駆動スレッド実行時の out-of-order スケジューリングを確認するのに主に用いられた．また，検証人月に関しては，基本動作および上記 1) に関してはそれほど費やさなかったが，上記 2) にその大半を費やした．ゆえに，CUE-v2 チップの設計難易度は，投機実行を行わない分，一般的な out-of-order スーパスカラよりも容易であると考える．

現段階での CUE-v2 チップの緒元を表 1 に示す．CUE-v2 チップは，cell-based 設計であり，Artisan 社がフリーライブラリプログラムにて提供する TSMC 社 0.18μm プロセス (CL018G) 用のスタンダードセルライブラリ (SAGE-X) を利用している．なお，HDL(Hardware Description Language) 記述は全て Verilog HDL により行っている．演算器に関しては，Synopsys 社の DesignWare Foundation を用い生成している．内蔵 SRAM に関しては，PGC 社が提供する BIST(Built-In Self-Test) メモリのハードマクロを利用している．PLL に関しては，Deskew 機能付きかつ逓倍クロックの生成可能なものが利用できなかったため，製作予定のボード上の FPGA の PLL よりクロックを供給することとし，TSMC 社が提供する Deskew 機能のみの PLL を利用した．DFT(Design For Test) に関しては，上述の BIST メモリに加え，19 本のスキャンチェーンを挿入しており，その coverage は 99.14%以上を予定している．また，表 1 中の消費電力は，ゲート数，活性率等から試作関係機関において見積もられた値である．

CUE-v2 のコア部の各モジュール毎のハードウェア量を表 2 に示す (四捨五入の関係上，合計値と各要素の総和は必ずしも一致しない)．本表では CUE-v2 チップの IO パッド，PLL，内蔵 RAM，および，CTS バッファを除いたいわゆるコア部の値を示している点に留意して頂きたい．また，本評価は，Synopsys 社の Design Compiler(2001.08-SP2) により論理合成を行った結果である．表 2 より，コア部の大部分 (62.4%) を動的

表 1 CUE-v2 チップ緒元

Process	TSMC 0.18 μm, 5Al 2Poly CMOS
Die Size	5.0 x 5.0 mm^2
Power Supply	1.8 V
Frequency	66 - 133 MHz
Power Dissipation	2.3 W @ 1.8 V 100 MHz
IO Pin Count	113
Package	Plastic BGA 256
Embedded RAM	Inst 32 KB, Data 32 KB

表 2 CUE-v2 のコア部のハードウェア量

Module	K gate (2NAND)	module/total(%)
IF0	22.2	5.1
IF1	6.0	1.4
ID0	3.5	0.8
ID1	10.0	2.3
FC	269.7	62.4
INT	21.0	4.9
LS	45.7	10.6
BR	1.1	0.3
SW	14.2	3.3
Register File	10.7	2.5
PCI Interface	26.8	6.2
Observe Circuit	1.1	0.3
Core Total	431.9	100.0

out-of-order スケジューリングを行う発火制御部 (FC) が占めることが分かる．このようになった原因として，2 種類のスレッドの実行に関して，CUE-v2 の実装が直接的かつ素直であることが挙げられる．具体的には，以下の 3 つが考えられる．1) データ駆動スレッド実行時のオーバフロー防止の為に，待ち合わせメモリのエントリ数が 64 と大きい．2)2 種類のスレッドで out-of-order スケジューリングに必要なオペランドの供給経路が異なることにより，Wakeup 回路中の連想記憶のポート数が増加してしまう．3)2 種類のスレッド間の資源競合回避を目的とした Select 回路の構成から規模の増大が免れ得ない．また，データ駆動，制御駆動のどちらか一方のみしか用いないモジュールは，BR の 0.3%，SW の 3.3%，Register File の 2.5%，ID1(半分以上がレジスタリードとリネーミングに用いられている) の 2.3%となっており，2 種類のスレッド間でハードウェアを比較的共有できているといえる．ただし，この共有のために各モジュールのハードウェア量が，通常のデータ駆動もしくはスーパスカラよりも増加している点には注意すべきである．

4. まとめと今後の課題

本論文では，データ駆動・制御駆動スレッドを命令レベルに同時・多重処理可能な，マルチメディアネットワーキング向きデータ駆動プロセッサ CUE-v2 の設計およびハードウェア規模について述べた．CUE-v2 は，逐次部の高効率化のために制御駆動スレッドのスーパスカラ的実行を可能としたため，フロントエンド部が従来のデータ駆動プロセッサよりも複雑化した．一方，out-of-order スーパスカラプロセッサから見れば，その表面的なハードウェアの追加はスレッド管理機構およびデータ駆動用の循環パス程度である．ただし，2 種類のスレッドの実行に関して，CUE-v2 は直接的かつ素直な実装をしているため，発火制御部が通常のスーパスカラと比較して大きくならざるを得なかった．したがって，今回の LSI 試作では，現状の発火制御機構の構成であっても規模および消費電力に関して問題は無い見通しであるが，今後 CUE-v2 のコアを複数搭載するチップマルチプロセッサ構成を実現する際には，その PE 数およびターゲット周波数によっては，更なる改良を加えた発火制御機構が必要であろう．また，CUE-v2 の性能評価に関しては，結果がまとまり次第，稿を改めて報告する予定である．なお，CUE-v2 は，2004 年中にテープアウトする予定である．

謝辞 本研究の一部は，STARC との共同研究によるものである．有益なコメントを頂いた STARC 平田雅規氏，シャープ 宮田宗一氏，木原誠一郎氏，日立 長坂充氏に感謝致します．また，本研究は東京大学大規模集積システム設計教育研究センターを通し，シノプシス株式会社およびケイデンス株式会社の協力で行われたものである．最後に，アセンブラの作成および動作検証に御助力頂いた我孫子泰祐氏，青木一浩氏に感謝致します．

参 考 文 献

1) A. Fox, S. D. Gribble, E. A. Brewer, and E. Amir, “Adapting to Network and Client Variability via On-Demand Distillation,” *Proc. of 7th ASPLOS*, pp. 160–170, 1996.
2) P. Crowley, M. Fiuczynski, J. Baer and V. Bershad, “Characterizing Processor Architectures for Programmable Network Interfaces,” *Proc. of 14th ICS*, pp. 54–65, May 2000.
3) T. Muramatsu, R. T. Shichiku, S. Miyata, and H. Nishikawa, “Super-Integrated Data-Driven Processors Realizing Hyper-Distributed System Environment,” *Proc. of 1998 PDPTA*, pp. 461-468, June 1998.
4) 西川, 青木, “プロトコル多重処理のデータ駆動型実現法とその実験的検討,” 信学論, vol. J85-D-I, no. 7, pp. 635–643, July 2002.
5) R. Kurebayashi, T. Takahashi, and H. Nishikawa, “A Data-Driven Implementation of Real-Time Video Compression,” *Proc. of 2002 PDPTA*, Vol. 3, pp. 1271–1274, June 2002.
6) R. A. Iannucci, “Toward A Dataflow/von Neumann Hybrid Architecture,” *Proc. of 15th ISCA*, pp. 131–140, June 1988.
7) S. Sakai, Y. Yamaguchi, K. Hiraki, Y. Kodama, and T. Yuba, “An Architecture of A Dataflow Single Chip Processor,” *Proc. of 16th ISCA*, pp. 46–53, June 1989.
8) 樽林, 伊藤, 高橋, 冨安, 西川, “逐次処理部のボトルネックの軽減と多重処理性能の維持を可能とするデータ駆動プロセッサ,” 信学論, vol. J87-D-I, no. 1, pp. 22–34, Jan. 2004.
9) D. M. Tullsen, et al., “Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor,” *Proc. of 23rd ISCA*, pp. 282–293, May 1996.
10) D. Burger and T. A. Austin, ”The SimpleScalar Tool Set, Version 2.0,” Technical Report CS-TR-97-1342, Univ. of Wisconsin-Madison, June, 1997.
11) S. Palacharla, N. P. Jouppi, and J. E. Smith, “Quantifying the Complexity of Superscalar Processors,” Technical Report CS-TR-96-1328, Univ. of Wisconsin-Madison, Nov. 1996.
12) M. Butler and Y. N. Patt, “An Investigation of the Performance of Various Dynamic Scheduling Techniques,” *Proc. of 25th MICRO*, pp. 1–9, Dec. 1992.