Vol.2013-SLDM-163 No.28
2013/11/28

サイクルタイム制約を考慮した低消費電力な束データ方式による非同期式 AVR プロセッサの設計

岩崎 翔太郎[1,a)]　齋藤 寛[1,b)]

概要：本稿では，要求スループットを満足するために，サイクルタイム制約を考慮しつつ低消費電力な束データ方式による非同期式 AVR プロセッサを設計する．初めに制御モジュールを定義し，このモジュールを用いてパイプライン制御回路を実現する．次にサイクルタイム制約を満足するために設計制約を生成し，標準的なライブラリやツールを用いて束データ方式による AVR プロセッサを合成する．また，レジスタファイルやメモリをグループ化し，それぞれ異なる書き込み信号で制御することによって低消費電力化を行う．実験では，設計された非同期式 AVR プロセッサの面積，実行時間，消費電力，消費エネルギーを評価し，同期式 AVR プロセッサと比較する．

Shotaro Iwasaki[1,a)]　Hiroshi Saito[1,b)]

Abstract: In this work, we design a low power asynchronous AVR processor with bundled-data implementation. The asynchronous AVR processor is designed considering a cycle time constraint so that it satisfies a throughout requirement. At first, we define a control module. Then, a pipeline control circuit is implemented using it. Second, we design the asynchronous AVR processor with bundled-data implementation using standard library and tools. In addition, we generate a set of design constraints for satisfying the given cycle time constraint. Finally, we propose a power optimization method. For power optimization, the register file and memories are grouped. Each group is controlled by a different write signal. In the experiments, we evaluate the asynchronous AVR processor in term of area, execution time, power consumption, and energy consumption comparing with the sychronous counter part.

1. はじめに

組込みシステムのマルチコア，メニーコア化によって，実行されるソフトウェアのボリュームが増していることから，組込みプロセッサに対して高性能で低消費電力なものが求められている．今日の多くの組込みプロセッサはグローバルクロック信号によって回路全体を制御する同期式回路だが，素子の微細化に伴い同期式回路は 2 つの問題に直面する．1 つ目はクロックスキューによる同期の失敗である．素子の微細化によって配線遅延が無視できなくなっており，クロックスキューが増大する傾向にある．これにより，データ転送における同期の失敗確率が増加する．2 つ目はクロックネットワークによる消費電力である．広範囲に高周波数なクロックが分配されることにより，クロックネットワークによる消費電力が増大してしまう．

非同期式回路は，グローバルクロック信号を使用せず，ローカルなハンドシェーク信号によって回路を制御するため，クロック信号に関連した問題が生じない．そのため，非同期式回路は潜在的に低消費電力であると言える．

これまでにもいくつかの組込み用途の非同期式プロセッサが設計されてきた．それらには Lutonium [1] や HT80C51 [2]，Pipelined Asynchronous 8051（PA8051） [3] 等があげられる．これらは Intel 社の 8051 プロセッサと互換性のある組込みプロセッサである．しかし，これらは動作仕様記述における構文と回路部品を 1 対 1 で変換する構文主導変換により設計しているため，設計制約を考慮した設計が難しい．また，最適化も限定的である．

本稿では，Atmel 社の AVR ATMega103 を対象に，要求スループットを満足するためにサイクルタイム制約を考慮しつつ低消費電力な非同期式 AVR プロセッサを設計する．また，制御モジュールを複製し，レジスタおよびメモリに対する書き込み信号を分割することによって，消費電力の最適化を行う．その後，標準的なライブラリやツールを用いて合成する．非同期式 AVR プロセッサの設計後，面積，実行時間，消費電力，消費エネルギーの評価を行い，同期式 AVR プロセッサと比較する．

本稿の構成は以下の通りである．2 節では束データ方式による非同期式回路について説明する．3 節では AVR プロセッサについて説明する．4 節では非同期式 AVR プロセッサの設計手法について説明する．5 節では実験結果について説明する．最後に 6 節では本稿のまとめと今後の課題を述べる．

[1] 会津大学
[a)] m5161140@u-aizu.ac.jp
[b)] hiroshis@u-aizu.ac.jp

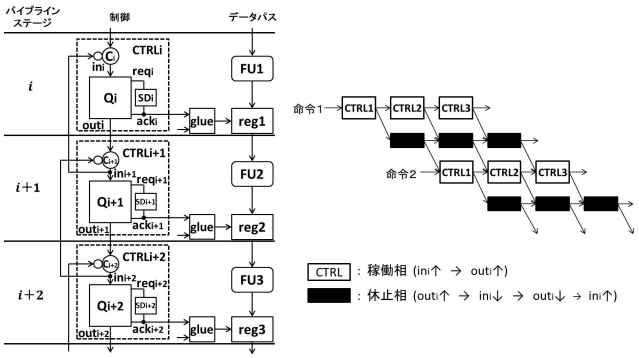

図 1　束データ方式による非同期式回路

2. 束データ方式による非同期式回路

2.1 回路モデル

束データ方式による非同期式回路は，非同期式回路の実装方式の一つで，N ビットのデータを N+2 本の信号で表す．図 1 左に本研究で用いる束データ方式による非同期式回路のパイプライン回路モデルを示す．データパス回路はレジスタや演算器，メモリより構成されていて，同期式回路と同じものを使用する．制御回路は制御モジュール $CTRL_i$ $(1 \leq i \leq m)$ より構成される．制御モジュール $CTRL_i$ は Q モジュール Q_i [5] および遅延素子 SD_i から構成される．各制御モジュールはパイプラインステージを制御し，各遅延素子 SD_i は演算の完了を保証する．

次に各制御モジュールの動作について説明する．制御回路は外部信号 start が 1 となると動作を始める．Q モジュール Q_i は，Q_{i-1} の out_{i-1} 信号が in_i 信号として入力することで動作を開始する．in_i 信号が入力した後，Q_i は要求信号 req_i を立ち上げる．req_i は遅延素子 SD_i を通り，ack_i を立ち上げる．ack_i が立ち上がると，req_i は立ち下がり，再び req_i は SD_i を通り，ack_i を立ち下げる．この時，ack_i はグルーロジックを通ることでレジスタへの書き込み信号へと変換される．ack_i の立ち下がり後，out_i が立ち上がり，in_{i+1} 信号として次の Q モジュール Q_{i+1} へと送られる．

この制御モジュール $CTRL_i$ には稼働相と休止相と呼ばれる 2 つの相が存在する．稼働相とは，制御モジュール $CTRL_i$ がデータパス回路を制御し，データパス回路で処理を行っている相のことである．休止相とは，制御モジュール $CTRL_i$ の初期化を行っている相のことで，このときデータパス回路は動いていない．in_i の立ち上がりから out_i の立ち上がりまでを稼働相，out_i の立ち上がりから out_i が立ち下がり in_i の立ち上がりまでが休止相である．制御モジュールは図 1 右の通り，この 2 つの相を繰り返して動作する．

2.2 タイミング制約

本稿で用いる束データ方式による非同期式回路は次の 3 つのタイミング制約を満たさなければならない．

2.2.1 セットアップ制約

レジスタの入力データは，レジスタのセットアップ時間よりも前に到達していなければならない．図 2 にセットアップ制約の例を示す．図 2 の実線はデータパス sdp を表し，このパスの最大遅延を T_{maxsdp} と表す．一方，破線は制御パス scp を表し，このパスの最小遅延を T_{minscp} と表す．また，レジスタのセットアップ時間を T_{setup}，T_{maxsdp} に対するマージンを sm と表す．この時，セットアップ制約は次式で表すことができる．

$$T_{minscp} > T_{maxsdp} * sm + T_{setup} \quad (1)$$

2.2.2 ホールド制約

レジスタにデータが書き込まれる際，入力データはホールド時間だけ維持されていなければならない．図 3 にホールド制約の例を示す．図 3 の実線は制御パス hcp を表し，この最大遅延を T_{maxhcp} と表す．一方，破線はデータパス hdp を表し，この最小遅延を T_{minhdp} と表す．また，レジスタのホールド時間を T_{hold}，T_{maxhcp} に対するマージンを hm と表す．この時，ホールド制約は次式で表すことができる．

$$T_{minhcp} > T_{maxhdp} * hm + T_{hold} \quad (2)$$

2.2.3 分岐制約

条件信号によって制御が分岐する場合，条件信号は分岐直前の制御モジュールからの出力信号よりも前に分岐判定論理に到達していなければならない．図 4 に分岐制約の例を示す．ここでは，図 4 内のレジスタ reg1 の出力信号が分岐判定に用いられるとする．この時，ack_i からレジスタ reg1 を通り，in_{i+1} 前の分岐判定論値の出力ピンまでの実線で表されたパス bdp の最大遅延を T_{maxbdp} と表す．一方，ack_i から 分岐判定論理の入力ピンまでの破線で表されたパス bcp の最小遅延を T_{minbcp} と表す．また，T_{maxbdp} に対するマージンを bm と表す．この時，分岐制約は次式で表すことができる．

$$T_{minbcp} > T_{maxbdp} * bm \quad (3)$$

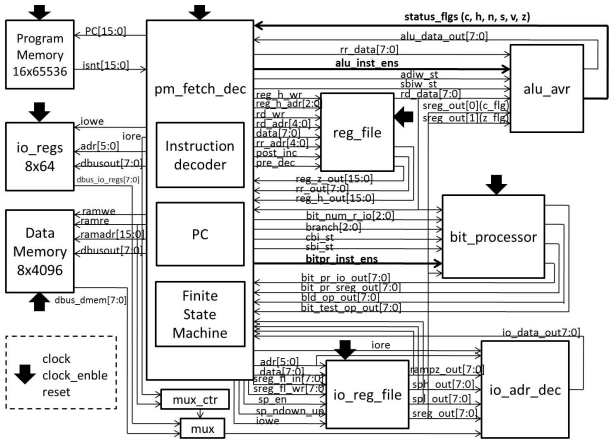

図 5　ATMega103 ブロック図

3. AVR プロセッサ

AVR プロセッサとは，Atmel 社によって開発された 8 ビットのマイクロコントローラである．組込みシステムに広く使用されており，センサーやモーター制御等に使用されている．

図 5 に本研究で用いる ATMega103 のブロック図を示す．

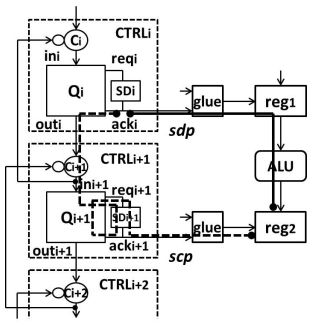

図 2　セットアップ制約に関するパス

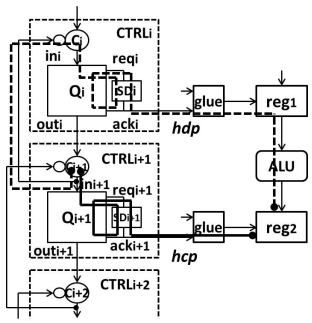

図 3　ホールド制約に関するパス

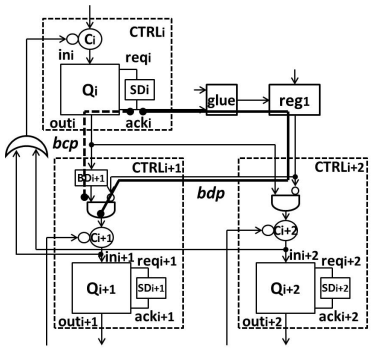

図 4　分岐制約に関するパス

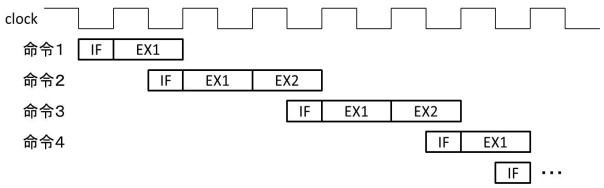

図 6　AVR プロセッサの動作

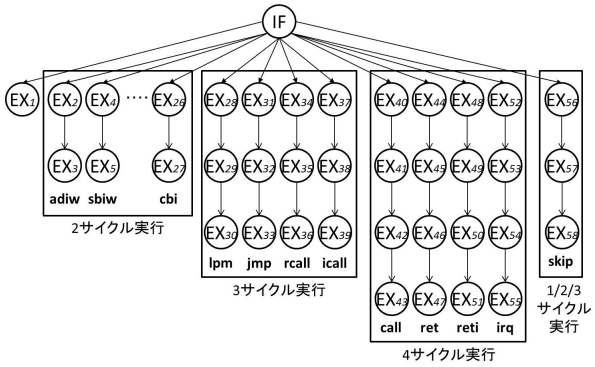

図 7　パイプライン制御のモデリング

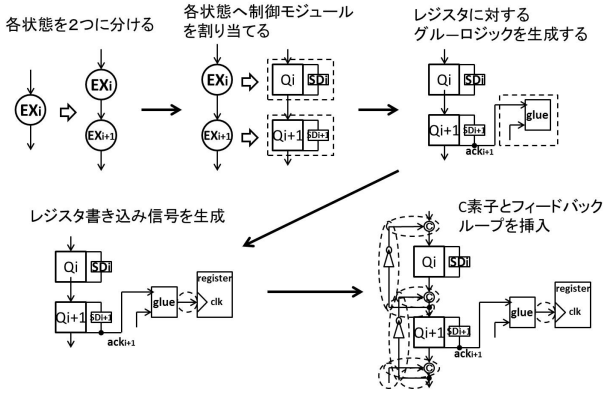

図 8　同期式から非同期式への変換方法

pm_fetch_dec は，フェッチされたプログラムのデコードや，プログラムカウンタや有限状態機械の管理を行い，主に他のモジュールの制御を行う．reg_file は，レジスタファイルを格納しており，8 ビットのレジスタを 32 個保有している．io_regs は，I/O 専用レジスタを格納しており，8 ビットのレジスタを 64 個保有している．alu_avr は，四則演算や論理演算の処理を行う．bit_processor は，ビット操作を行う．io_regs は，ステータスレジスタやスタックポインタ等の特殊なレジスタを保有している．io_adr_dec は，各モジュールから受け取ったデータの中から必要なデータのみを選択し，pm_fetch_dec へ送る．

図 6 に AVR プロセッサの動作を示す．AVR プロセッサは 2 段のパイプラインとなっており，IF ステージと EX ステージから成る．IF ステージではプログラムメモリから命令をフェッチする．EX ステージではフェッチされた命令に応じた動作を行う．EX ステージがマルチサイクル実行の命令も存在し，2 から 4 サイクル必要とする．EX ステージの最後のサイクルで，次の命令の IF ステージが実行される．

4. 設計手法

本研究では，要求スループットを満足するために，目標サイクルタイムを与え，ATMega103 を非同期式プロセッサとして設計する．また，レジスタやメモリに対する書き込み信号を分割することによって消費電力を抑える低消費電力化手法を提案する．

4.1 ATMega103 の非同期化

ここでは，ATMega103 の Verilog HDL モデルから非同期式の Verilog HDL への変換方法について説明する．本稿で使用する ATMega103 は OpenCore [6] より取得した．まず，図 7 のように有限状態機械（FSM）を用いて，ATMega103 の命令毎にパイプライン制御のモデリングを行う．次に，図 8 の変換手順で FSM モデルから非同期式制御回路を生成する．

はじめに，FSM の各状態を 2 つに分割する．分割後，制御モジュール $CTRL_i$ を各状態へと割り当てる．レジスタまたはメモリが複数の制御モジュールによって制御される場合には，レジスタまたはメモリへの書き込み信号を生成するグルーロジックを生成する．その後レジスタ書き込み信号はレジスタへ，メモリ書き込み信号はメモリへ割り当てる．最後に，C 素子およびフィードバックループを追加する．

ここで，変換手順の中で各状態を 2 つに分割する理由について説明する．もし各状態が分割されなかった場合，図 1 右のように休止相の存在によりすべてのパイプラインステージを同時に実行することができない．すなわち，同期式回路のパイプライン動作と同様の動作を行うことができない．結果，サイクルタイムが同期式と比較した際に増大

してしまい，要求サイクルタイムを満足することが難しくなってしまう．これを解決するために，各状態を 2 つに分割する．分割することにより休止相が隠蔽され，すべてのパイプラインステージを同時に実行することが可能となり，要求サイクルタイムを満足しやすくなる．なお，この分割は制御回路だけのもので，データパス回路には変更を加えない．

4.2 制約生成

非同期式回路にはグローバルクロック信号がないので，目標サイクルタイムを実現するために，レジスタの書き込み信号（ローカルクロックと呼ぶ）やパス毎に最大遅延制約を与える．なお，ここではサイクルタイム制約を CT とする．

まず，レジスタへの書き込み信号となるローカルクロック ack_i には，create_clock コマンドを用いる．ローカルクロック制約の値は $CT \times \alpha$（$0 < \alpha \leq 1$）とし，α は設計者が決める．

次に，制御回路にはローカルクロック信号が存在しないので，パス毎に set_max_delay を用いて最大遅延制約を生成する．最大遅延制約の値は $CT \times \beta$（$\alpha < \beta \leq 1$）とし，β も設計者が決める．なお，$\alpha < \beta$ とするのは，セットアップ制約を満足しやすくするためである．

また，制御回路の最適化を防ぐために，制御モジュール毎に set_dont_touch コマンドを用いて非最適化制約を与える．この制約を与えない場合，制御回路が最適化されてしまい，予期せぬ動作が起きてしまう．

4.3 設計フロー

図 9 に本稿で用いる設計フローを示す．まず，非同期式 AVR プロセッサの Verilog HDL モデルと制約を準備する．次に，論理合成およびレイアウト合成を行う．また，この時 STA で用いる解析コマンドを一緒に生成する．その後，タイミング解析およびタイミング検証を行う．タイミング検証時に，2.2 節で解説したタイミング制約に違反したパスがある場合には，遅延調整用のコマンドを生成し再びレイアウト合成，タイミング解析およびタイミング検証を行う．これをすべてのタイミング制約を満足するまで行う．設計完了後（すべてのタイミング制約を満足させた後），比較および性能の評価を行う．

4.3.1 論理合成とレイアウト合成

Verilog HDL モデル，制約，および標準的なテクノロジーライブラリを論理合成ツールに与え，論理合成を行う．論理合成では，ゲートレベルネットリスト，および入力として与えた制約を 1 つのファイルにまとめた synopsys design constraint（SDC）ファイルを生成する．

論理合成後，出力されたゲートレベルネットリストおよび制約，テクノロジーライブラリをレイアウト合成ツールに与え，レイアウト合成を行う．レイアウト合成では，ゲートレベルネットリスト，および standard delay format（SDF）を生成する．

4.3.2 静的タイミング解析およびタイミング検証

レイアウト合成後，出力されたゲートレベルネットリストおよび SDF ファイルを用いて静的タイミング解析（Static Timing Analyzer, STA）およびタイミング検証を行う．STA ツールでは report_timing コマンドを用いて，

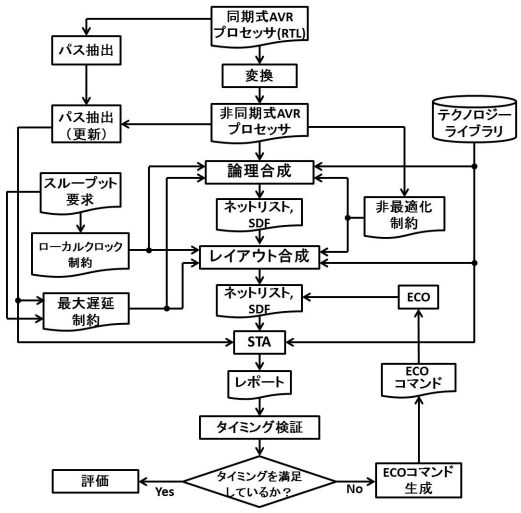

図 9 設計フロー

2.2 節で解説したタイミング制約を検証するために必要となるパスの最大遅延および最小遅延を求める．

セットアップ制約における制御回路のパス scp に対する report_timing コマンドは以下のように生成する．制御モジュール $CTRL_i$ の Q_i の ack_i から 次の制御モジュール $CTRL_{i+1}$ の Q_{i+1} の ack_{i+1} までの最小遅延を解析したいので，ack_i から C_{i+1}，C_{i+1} から SD_{i+1}，SD_{i+1} から Q_{i+1} 内のラッチ，Q_{i+1} 内のラッチから SD_{i+1}，SD_{i+1} からレジスタのクロックピンまで，とパスを 5 つに分割し report_timing コマンドを生成する．なおパスを分割する理由は，各制御モジュール内にラッチが含まれており，それによってパスが分断されてしまうからである．なおラッチを含まなければ，組み合わせループが発生し，STA ツールは適当な所でループを断ってしまうため，正しい解析を行えなくなる．データパス回路のパス sdp に対しては，$CTRL_i$ の ack_i からレジスタのクロックピンまでと ack_i によって制御されるレジスタから ack_{i+1} によって制御されるレジスタまでのデータパスの最大遅延を解析するための report_timing コマンドを生成する．同様にホールド制約や分岐制約に関するパスに対しても report_timing コマンドを生成する．

タイミング検証は，2.2 節の各タイミング制約の制約式（1），（2），（3）における差（左辺－右辺）を計算することで行う．この差が負の場合は，タイミング違反ということになる．

4.3.3 遅延調整

タイミング制約に違反しているすべてのパスに対して，タイミング制約を満足するように遅延調整を行う．調整はホールド，分岐，セットアップ制約違反の順で行う．この順番は，ホールド制約違反や分岐制約違反による調整がセットアップ制約に影響を与えることに起因する．

次に遅延調整の仕方について説明する．まず，ホールド制約に違反した際には，該当するレジスタのデータ入力ピンの前に制約式の差の絶対値に相当するバッファを挿入する．分岐制約に違反した際には，該当する BD_i に制約式の差の絶対値に相当するバッファを挿入する．セットアップ制約に違反した際には，該当する $CTRL_i$ に関連するセットアップ制約のうち違反しているパスの中で，制約式の差の絶対値の最大値を 4 で割ったものに相当するバッファを

```
addInst –cell buf –inst CTRLi/SDi/d5
addNet CTRLi/Sdi/w4
attachTerm CTRLi/SDi/d4  O  CTRLi/SDi/w4
attachTerm CTRLi/SDi/d5  I  CTRLi/SDi/w4
attachTerm CTRLi/SDi/d5  O  CTRLi/SDi/out
```

図 10　セットアップ違反のための ECO コマンド

SD_i へ挿入する．4 で割る理由は，レジスタにデータを書き込むまでにパイプライン制御が 2 分割され，それぞれにおいて SD_i を 2 度通るためである．

一方，SD_i に過剰にバッファが入って $CTRL_i$ によって制御される全てのレジスタのセットアップ制約が満たされた場合，すなわち制約式の差が正となった場合には，$CTRL_i$ によって制御されるデータパスの中で，制約式の差の最小値を 4 で割ったものに相当するバッファを SD_i から削除する．

実際の遅延調整は，レイアウト合成ツールにおける ECO にて行う．そのため，図 10 のような ECO コマンドを生成する．この例では，SD_i の最後のバッファ d4 の後ろに新しくバッファ d5 をつなげることで，遅延を増加させている．

4.4　消費電力最適化手法

本稿では，レジスタファイルおよびメモリの書き込み信号を分割することによって動的な電力消費を削減する手法を提案する．通常，AVR のレジスタファイルおよびメモリは 1 つの書き込み信号（グローバルクロック信号）によって制御される．しかしファンアウトが多ければ，大量のバッファが挿入されてしまい，結果として消費電力が増大してしまう．そこで，レジスタファイルやメモリをアドレスに応じてグループ化し，制御モジュールを複製することで，レジスタやメモリへの書き込み信号を分割する．これにより，書き込み信号のファンアウト数が減少し，挿入されるバッファも減少するため，電力消費の削減を行うことが期待できる．

分割の方針は以下の通りである．まずレジスタファイルおよびメモリを先頭アドレスから 2 の n 乗のサイズで割り切れるように分ける．次に，グループの数に応じて制御モジュールを複製する．複製された各制御モジュールは決められたグループのみを制御する．最後に，レジスタファイルおよびメモリのアドレスによって制御を分割させるために，制御モジュール間にグル―ロジックを挿入する．図 11 は書き込み信号分割の例を示す．この例では，レジスタファイルを 4 つにグループ化し制御モジュールを 4 つ複製し，書き込み信号を 4 分割している．

5.　実験

実験では，非同期式 AVR プロセッサを設計し，面積，実行時間，消費電力，消費エネルギーの評価を行う．また，同期式 AVR プロセッサと比較を行う．

今回の実験では，同期式 AVR プロセッサ（sync），非同期式 AVR プロセッサ（async），レジスタファイルをグループ化した非同期式 AVR プロセッサ（async_x_0_0（x = 2,4,8)），プログラムメモリをグループ化した非同期式 AVR プロセッサ（async_0_y_0（y = 4,8,16)），データメモリをグループ化した非同期式 AVR プロセッサ（async_0_0_z（z = 2,4,8)），および，それぞれのグループ化で良かったもの同士を組み合わせた非同期式 AVR プロセッサ（async_4_4_2），を設計した．x,y,z はそれぞれの分割数を表している．なお，レジスタファイルは 8 ビットのレジスタ 32 個からなり，プログラムメモリは 16 ビットでワード数は 65,536 である．データメモリは 8 ビットでワード数は 8,192 である．設計の方針として，クロック周波数が最高となる同期式 AVR プロセッサを探索し．その時のクロックサイクルタイム 7ns をサイクルタイム制約とした．また，create_clock や set_max_delay に対する α, β をそれぞれ 0.95, 0.98 とした．

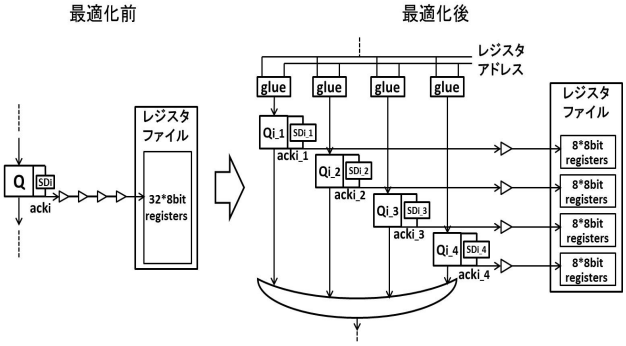

図 11　レジスタファイルへの書き込み信号を 4 分割

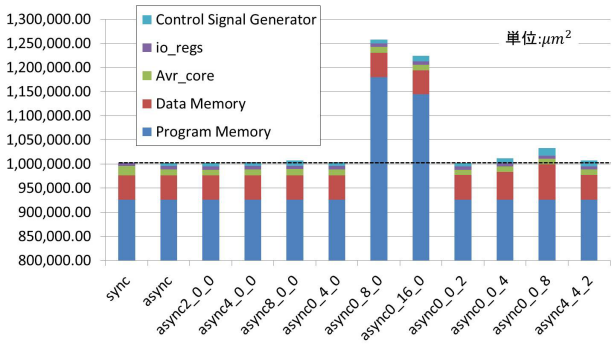

図 12　回路全体の面積

設計に際して，以下のツールおよびテクノロジを使用した．論理合成には Synopsys 社の Design Compiler を，レイアウト合成には Cadence 社の EDI を，タイミング解析には Synopsys 社の PrimeTime を，シミュレーションには Synopsys 社の VCS を使用した．テクノロジには eShuttle 65nm を使用した．また，テストベンチには微分方程式を解く C プログラムを用い，IAR 社の IAR Embedded Workbench for Atmel AVR [7] を用いてコンパイルすることで得られる HEX ファイルを利用した．

図 12 は回路全体の面積を表す．面積は EDI の reportGateCount コマンドより取得した．図から async0_8_0, async0_16_0, async0_0_8 以外は，すべて sync と同等の面積であることがわかる．これら 3 つの AVR の面積が増大した理由は，メモリのサイズにある．グループ化に伴いメモリ自体も分割したが，メモリのワード数を半分にしても面積は半減しなかったことに起因する．(16 ビットでワード数が 16,384 のメモリの面積は 231499.2μm^2，ワード数が 8,192 のメモリの面積は 147470.6μm^2，ワード数が 4,096 のメモリの面積は 71510.5μm^2)．なお，図 12 内の

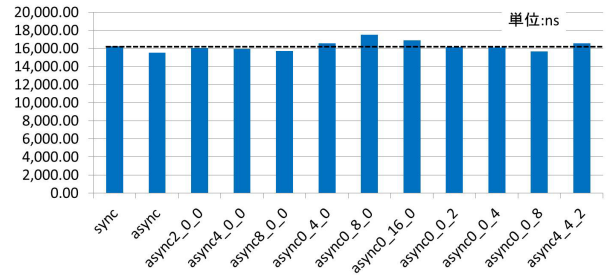

図 13 実行時間

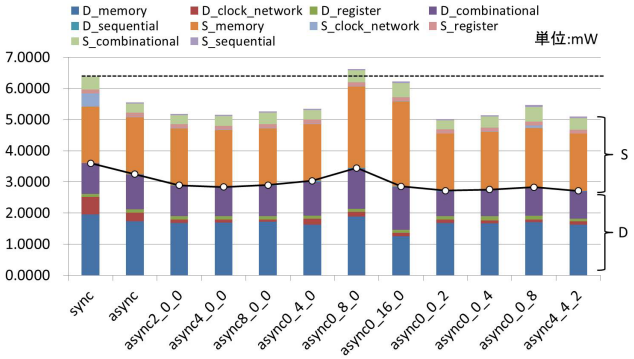

図 14 消費電力

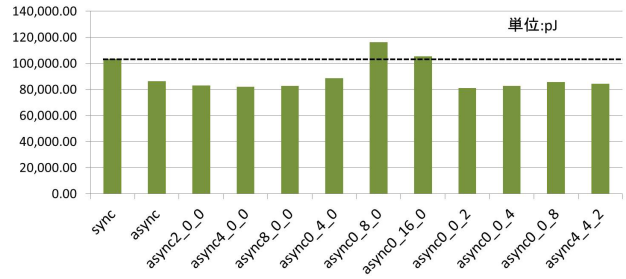

図 15 消費エネルギー

ControlSignalGenerator は非同期式制御回路を表し，全体の 1%にも満たない．このことから，非同期化に伴う面積のオーバーヘッドは小さいことがわかる．

図 13 はシミュレーションより得た実行時間を表す．実行時間に関してはプログラムメモリをグループ化したものを除き，すべて sync と同等の実行時間となった．これは，サイクルタイムを考慮し設計を行ったためである．プログラムメモリをグループ化したものが悪くなった理由は，グループ化により制御モジュールを分割し，グルーロジックを挿入した結果，遅延が伸びたことに起因する．

図 14 はシミュレーションを基にした PrimeTime による電力解析の結果を表す．D_memory のように D で始まるものは動的電力（Dynamic power）を表し，S_memory のように S で始まるものは静的電力（Static power）を表す．

消費電力に関しては，async0_8_0 以外は，すべて sync と比べて良い結果となった．最も良かったのは async0_0_2 で，sync と比較して約 21%の削減である．レジスタファイルやメモリの書き込み信号の分割を行わなかった async の場合は，sync と比べ約 13%の削減である．また，async と比較して，async0_8_0，async0_16_0，async0_0_8 以外は電力を削減できたということより（async に対して約 4%から約 10%の削減），提案した書き込み信号の分割による動的電力の削減は多くのケースで効果があることがわかる．動的電力だけで見てしまえば，async0_16_0 と async0_0_8 にも効果があることが分かる．一方，async0_8_0，async0_16_0，async0_0_8 の全体の消費電力が async より良くならなかった理由は，メモリ自体を分割したことによる面積の増加，それに伴う静的電力の増加に起因する．今後は電力モデルを用いた最適な分割手法を検討する予定である．

図 15 は消費エネルギーを表し，実行時間と消費電力の積である．消費エネルギーに関しては消費電力同様，async0_0_2 が最も良い結果となり，81,007.19pJ であった．これは sync と比べ，約 22%の消費エネルギーの削減である．

6. 結論

本稿では，サイクルタイムを考慮した低消費電力な AVR プロセッサを設計した．また，レジスタファイルおよびメモリをグループ化し，書き込み信号を分割することによって，動的な消費電力を削減する手法を提案した．実験の結果より，グループ化を行うことで非同期式 AVR プロセッサの消費電力および消費エネルギーを同期式 AVR プロセッサと比較し，それぞれ最大で 21%と 22%削減できた．

今後は，レジスタファイルやメモリの最適なグループ化を決定する手法の提案，およびさらなる低消費電力化のための，制御モジュールの共有，パワーゲーティング，オペランドアイソレーションといった他の消費電力最適化手法の実装を行う．また，様々なテストベンチを用いて同期式 AVR プロセッサと比較する．

謝辞 本研究は，東京大学大規模集積システム設計教育センターを通し，シノプシス株式会社およびイー・シャトルの協力で行われたものである．また，本研究は科研費若手研究（B）24700051 の助成による．最後に本研究を遂行するにあたって助言を下さった飯塚成氏に感謝申し上げたい．

参考文献

[1] A.J.Martin, M.Nystrom, K.Papadantonakis, P.I.Penzes, P.Prakash, C.G.Wong, J.Chang, K.S.Ko, B.Lee, E.Ou, J.Pugh, E.V.Talvala, J.T.Tong, and A.Tura, “The Lutonium: a sub-nanojoule asynchronous 8051 microcontroller,” Proc. ASYNC, pp. 14-23, May 2003

[2] Handshake Solutions, “HT80C51,” http://www.handshakesolutions.com/.

[3] C.J.Chen, W.M.Cheng, R.F.Tsai, H.Y.Tsai, and T.C.Wang, “A Pipelined Asynchronous 8051 Soft-core Implemented with Balsa,” Proc. APCCAS, pp. 976-979, Nov. 30 2008-Dec. 3 2008

[4] J.Sparso, and S.Furber, ”Principles of asynchronous circuit design: a systems perspective.,” Springer, 2001

[5] F.U.Rosenberger, C.E.Molnar, T.J.Chaney, and TP.Fang, “Q-Modules: Internally Clocked Delay Insensitive Modules,” IEEE Transaction of Computer, pp. 1005-1018, Volume: 37, Issue: 9, Sep. 1988

[6] Open Core, http://www.opencores.org/

[7] IAR Embedded Workbench for Atmel AVR, http://www.iar.com/