

博士論文

メモリをベースとしたマイコン用
再構成可能デバイスとその応用に関する研究

**Memory-based reconfigurable device for
microcomputers and its application**

金沢大学大学院 自然科学研究科
電子情報科学専攻

学籍番号：1424042014

氏名：川村 嘉郁

主任指導教員名：松田 吉雄

2017年9月

目次

第1章	序論	1
1.1	研究の背景	1
1.1.1	マイコンの歴史的背景	2
1.1.2	マイコンの市場動向	5
1.2	研究の目的	6
1.3	本論文の構成	7
第2章	プログラマブルロジックデバイスのマイコン搭載への課題	11
2.1	緒言	11
2.1.1	マイコンの現状と今後の課題	11
2.1.2	プログラマブルロジックデバイス技術動向	14
2.1.3	マイコンとプログラマブルロジックデバイスの課題	17
2.2	マイコン搭載プログラマブルロジックデバイスアーキテクチャの探索	18
2.2.1	プログラマブルロジックデバイスの構造分析	18
2.3	マイコンアーキテクチャの比較	22
2.4	結言	23
第3章	FPSM アーキテクチャ	25
3.1	緒言	25
3.2	基本論理素子 PMU アーキテクチャ	25
3.2.1	基本論理素子の検討	25
3.2.2	コンセプトと課題	25
3.2.3	カウンタ/タイマ機能を実装する基本論理素子モデルの検討	26
3.3	PMU のマイクロプログラム制御方式	30
3.3.1	PMU のマイクロ命令	31
3.3.2	PMU シミュレーションモデル	36
3.4	FPSM アーキテクチャの概要	39
3.4.1	PMU アレイ構成	40
3.4.2	MCU インタフェース	41
3.4.3	FPSM のメモリ管理とアクセス方法	42
3.5	スイッチボックス (SB)	44
3.6	結言	48
第4章	FPSM モデルシミュレーションと FPGA 実装による評価	50
4.1	緒言	50
4.2	PMU モデルと各周辺回路シミュレーション	50
4.2.1	基本論理演算のモデリングと評価	51
4.2.1.1	カウンタモデル	55
4.2.1.2	シフトレジスタモデル	61

4.2.1.3	演算器モデル	64
4.2.2	マイコン周辺回路のモデリングと評価	68
4.2.2.1	FIFO モデル	68
4.2.2.2	シリアル通信インタフェースモデル	71
4.2.2.3	PWM モデル	78
4.2.3	PWM の FPGA 実装	80
4.3	結言	83
第5章	実験チップの試作と評価	85
5.1	緒言	85
5.2	FPSM の論理合成	85
5.3	実装設計	85
5.4	試作と評価	87
5.4.1	周辺回路機能の実装評価	88
5.4.2	消費電力測定	91
5.4.3	shmoo plot 評価	91
5.5	FPGA との比較	92
5.6	結言	93
第6章	パケットフィルタ応用	95
6.1	緒言	95
6.2	パケット検索方式	95
6.3	一致/不一致検出パケット検索エンジン	96
6.3.1	不一致検出回路	96
6.3.2	一致検出回路	97
6.4	スループット評価	100
6.5	TEG チップと評価結果	102
6.6	結言	105
第7章	結論	107
7.1	基本論理素子 PMU アーキテクチャ	107
7.2	FPSM アーキテクチャ	107
7.3	応用展開	108
7.4	今後の課題と展望	108
謝辞	110
業績目録	111

記号・略称の解説

記号・略号	全文	意味
ALU	Arithmetic Logic Unit	算術演算や論理演算処理を行う装置で、マイクロプロセッサの構成要素の一つ
ASIC	Application Specific Integrated Circuit	特定用途向け集積回路。特定ユーザの用途に特化したカスタム製品
ASSP	Application Specific Standard Product	特定用途向け標準品。分野・用途を限定し、機能・目的を特化させた汎用製品
BLE	Basic Logic Element	論理ブロックの基本要素。LUT、FF およびセレクタで構成される
CAM	Content Addressable Memory	連想メモリ。主にネットワーク機器の packets 検索等に用いられる
CPLD	Complex Programmable Logic Device	SPLD の AND-OR アレイ構造を複数プログラマブルスイッチで結合した構成の PLD
CPS	Cyber Physical System	現実世界から得られるデータを収集し、これらデータを処理、活用することで、あらゆる社会システムの効率化、新産業の創出、知的生産性の向上を図る概念
CPU	Central Processing Unit	中央処理装置
CRC	Cyclic Redundancy Check	誤り検出方式の一つ。データ値をある定数で割った余り（余剰）を用いて誤り検出を行なう
DRAM	Dynamic Random Access Memory	ダイナミック・ランダム・アクセス・メモリ、一定時間毎に記憶保持のための再書き込み（リフレッシュ）が必要、電源を落とすと記憶内容は消去される
DRP	Dynamic ReConfigurable Processor	動的再構成可能プロセッサ、ノイマン型の ALU アレイで構成され、並列動作が可能
DSP	Digital Signal Processor	デジタルシグナルプロセッサ、信号処理で多用される積和演算を高速実行する
EEPROM	Electrically Erasable Programmable Read Only Memory	不揮発性メモリの一種、ユーザによって電氣的に消却・再プログラム可能な ROM
EPROM	Erasable Programmable Read Only Memory	不揮発性メモリの一種、ユーザによってプログラム可能な ROM、紫外線で一括消去が可能
FeRAM	Ferroelectric Random Access Memory	強誘電体メモリ、強誘電体のヒステリシス(履歴効果)を利用した不揮発性 RAM
FF	Flip-Flop	順序回路の基本要素。1 ビットの情報を一時的に"0/1"の状態として記憶する論理回路

FIFO	First In First Out	先に書き込んだものを先に取り出すバッファ動作
FPGA	Field Programmable Gate Array	LUT を基本論理素子とし、これらをアレイ状に配置した PLD
HDL	Hardware Description Language	回路の設計，構成を記述するハードウェア記述言語
IoT	Internet of Things	モノのインターネット，様々な「モノ」がインターネットに接続され，これらの情報を利用する仕組み
IP	Intellectual Property	他にライセンス供与する目的で準備された知的財産権のある回路設計データ等
LSB	Least Significant Bit	2進数で表現されたデータ列の最下位ビット
LUT	Look Up Table	所望の関数の真理値表をメモリに保持し，必要に応じて参照することで組み合わせ回路を実現する
MCU	Micro controller unit	CPUに加えて，ROMやRAMなどのメモリ，入出力や通信用のポート，タイマ，ADコンバータといった周辺機能までを1チップ上に集積したマイクロコンピュータまたはマイコン
MPU	Micro processor unit	マイクロプロセッサユニット，CPUを構成要素としたLSI
MRAM	Magnetoresistive Random Access Memory	磁気抵抗メモリ，スピントロニクス of GMR 効果（Giant Magneto Resistive effect：巨大磁気抵抗効果）を利用した不揮発性RAM
MSB	Most Significant Bit	2進数で表現されたデータ列の最上位のビット
OTPROM	One Time Programmable ROM	1回のみ書き込み可能なROM，ヒューズROM，消去窓無EPROMなどがある
PC	Program Counter	次に実行する命令が格納されているメモリ上のアドレスを記憶するレジスタ
PLA	Programmable Logic Array	プログラマブルロジックデバイスの呼称のひとつ
PLD	Programmable Logic Device	プログラマブルロジックデバイスの総称
PWM	Pulse Width Modulation	パルス幅変調，特定の波長周期のパルス幅（デューティ比）を変化させて変調する方式
RAM	Random Access Memory	メモリ内の読み書き（ランダムアクセス）がどこに記録されたデータでも同じ時間で実行可能なメモリ

ReRAM	Resistive Random Access Memory	抵抗変化型メモリ，電圧印加による電気抵抗の変化を利用した不揮発性 RAM
ROM	Read Only Memory	読み出し専用のメモリ，マスク ROM など
RTL	Register Transfer Level	レジスタ転送レベル，論理回路をハードウェア記述言語で記述する際の設計抽象度のレベル
SCI	Serial Communication Interface	調歩同期式のシリアル通信方式 UART (Universal Asynchronous Receiver/Transmitter) と呼ばれる
SCK	Serial Clock	同期式のシリアル通信で利用するクロック
SoC	System on Chip	所望の装置やシステムの動作に必要な機能を，一つの半導体チップに実装した LSI
SPLD	Simple Programmable Logic Device	AND-OR アレイ (プロダクトターム) で構成される比較的規模の小さい PLD の総称
SRAM	Static Random Access Memory	RAM の一種で，データを定期的に取り直し (フレッシュ) が不要
TAT	Turn Around Time	半導体製品の製造着工から完成までの期間

第1章 序論

1.1 研究の背景

一般に、マイクロコンピュータ（以下、マイコン）はCentral Processing Unit（CPU）を搭載した演算処理を行う半導体デバイスである。これらは汎用向け Micro Controller Unit（MCU）と高性能/高機能用途向け Micro Processor Unit（MPU）に大別されているが、総称してマイコンと呼ばれている。このマイコンの歴史は、1971年に電卓向けにインテル社が4ビットのマイクロプロセッサ Intel 4004を開発・製品化されたことに始まる。ユーザのカスタム品として製品化され、演算部の4ビットマイクロプロセッサ（4004）、データメモリ用RAM（4002）、プログラム用ROM（4001）および周辺回路・I/Oポート（4003）の4種類のチップで電卓システムが構成されていた[1-1][1-2][1-3]。これらはユーザの開発費で開発されたカスタム製品であったが、ユーザの値下げ要求と引き換えにインテル社が外販権を獲得し、MCS-4として一般にも販売されることとなった。これは現在もマイコンビジネスのモデルの一つとなっている。当時、マイクロプロセッサ Intel 4004は10 μ mの製造プロセスを用い、チップサイズは幅約3mm、長さ約4mmの大きさで、約2300個のトランジスタが集積され、動作周波数は108kHz、1命令の処理時間は約10.8 μ sであった [1-4]。

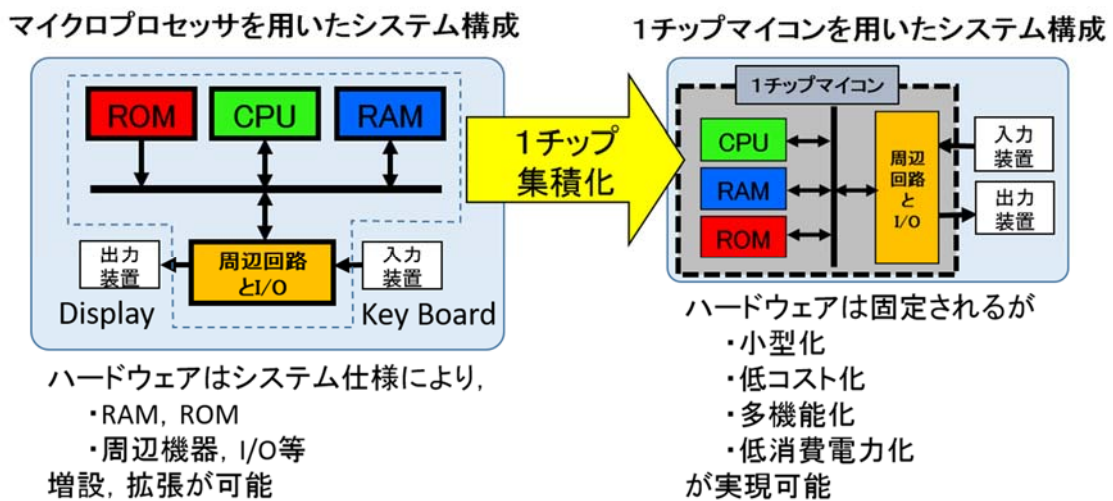


図 1-1 シングルチップマイコン

インテル社のMCS-4がCPU、ROM、RAMおよび周辺回路・I/Oポートのチップセットとして販売されてからはCPU、ROM/RAM、カウンタ/タイマ、シリアル通信、A/Dコンバータ等を一つのチップ上に集積したシングルチップマイコン化が進められ（図1-1）、以降様々なシングルチップマイコン製品が登場した。

MCS-4のようなチップセットは、ROM/RAM容量の増加やターゲットシステム構築に必要な周辺機能、I/O等の拡張性は高いが、小型化には向いていない。特に組み込み機器向けには、小型、低コスト化および低消費電力化を図るため、積極的にシングルチップマイコンが製品化された。

当時はシステム規模も小さく、シングルチップマイコンにはシステム設計に必要なハードウェア部分

が揃っており，ソフトウェア部分を置き換えるだけで様々なシステムに対応でき，ソフトウェアプログラムがもたらす開発期間の短縮は最大のメリットであった．システムをハードウェアのみで開発する場合，開発途上でハードウェアの機能/仕様の変更が発生した場合の手間と時間は多大なものであり，ソフトウェアで機能/仕様を変更できるメリットは大きい．これはソフトウェアで機能実装/仕様変更できるシングルチップマイコンのユーザアプリケーションへの適応性の高さを示している．

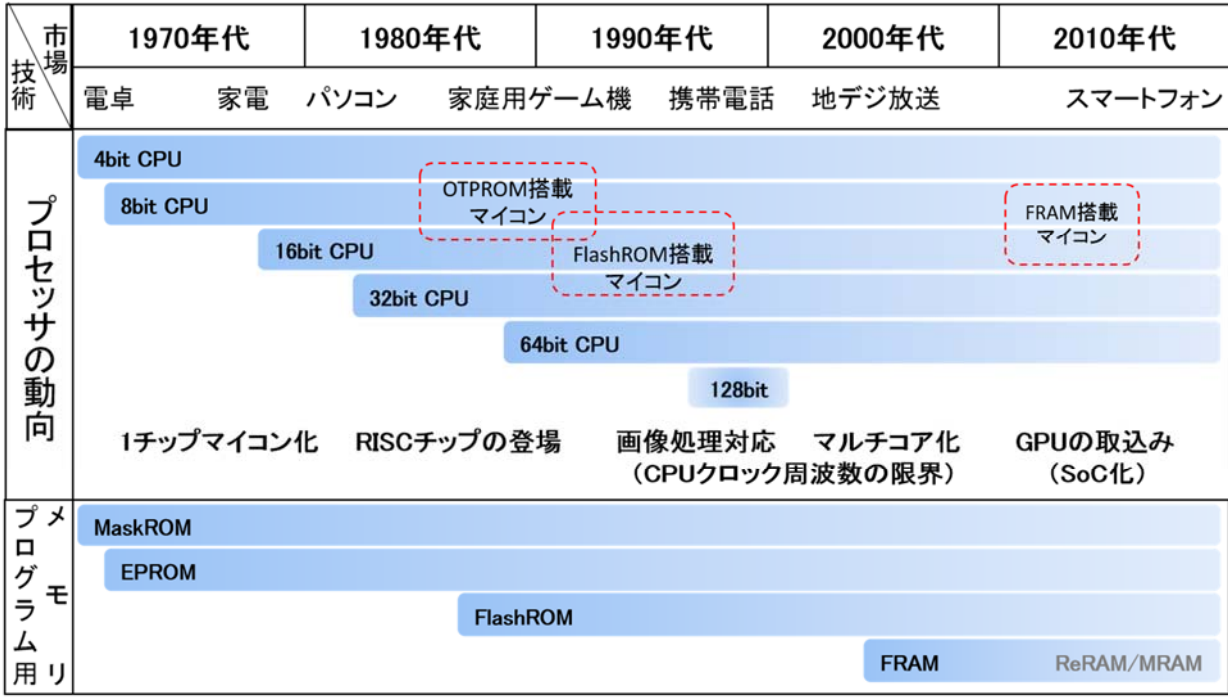


図 1-2 CPU とプログラム用メモリの変遷

1. 1. 1 マイコンの歴史的背景

図 1-2 に CPU とプログラム用メモリの変遷を示す．1970 年代から 1980 年代にかけて CPU の 8/16/32/64 ビットへのワード長拡張とともに，動作周波数も数百 KHz から数十 MHz へと引き上げられ，演算性能も Intel 4004 の 50 倍程度に向上が図られた．さらに実装技術の向上により，シングルチップマイコンに内蔵する ROM/RAM 容量の増加，周辺回路のバリエーションおよび I/O 数も増加した．1990 年代には，パソコンや組み込み機器向けに最適化設計された 32 ビット CPU をベースとした CISC/RISC アーキテクチャを持つマイコンが各社から発表された．

1990 年代後半になると高性能マイコンはゲーム機器，デジタル放送向けマルチメディア対応のための音声・画像処理技術への応用が盛んになるとともに，CPU の動作周波数の限界もさげばれ始めた．このため，CPU と Digital Signal Processor (DSP) を組み合わせた製品，CPU と専用アクセラレータを組み合わせたヘテロジニアス構成の Application Specific Standard Product (ASSP) や System on Chip (SoC) が登場した．これらは携帯電話の音声信号処理や 2000 年前半にサービスが始まった携帯電話に搭載されるテレビ電話機能や地上波テレビ受信等に用いられた．

このように，1970 年後半から 2000 年代前半にかけて様々なマイコンおよび CPU を搭載する ASSP/SoC が開発・製品化された．これらはユーザの組み込み機器システムの要求仕様に合わせて ROM/RAM の増設，

周辺機能および I/O の拡張が盛んに行われた。

2000 年代に入ると、市場に改めて 64 ビット CPU が投入され、CPU の動作周波数も GHz 帯となった。さらにホモジニアス構成のデュアルコア、クワッドコア等の CPU マルチコア搭載製品が登場し、動作周波数は 3GHz を超えるまでになった。これらマイコン製品はゲーム機器、携帯電話・スマートフォン、さらにはパソコンやインターネット通信サーバ等の比較的大きいシステムにも利用されてきた。この中でも、特にシングルチップマイコンは集積化技術の発展とともに小型化、低コスト化、多機能化および低消費電力化を実現し、汎用マイコン製品としてコンシューマ、産業機器や車載機器等の組み込み機器等に利用され、発展してきた。

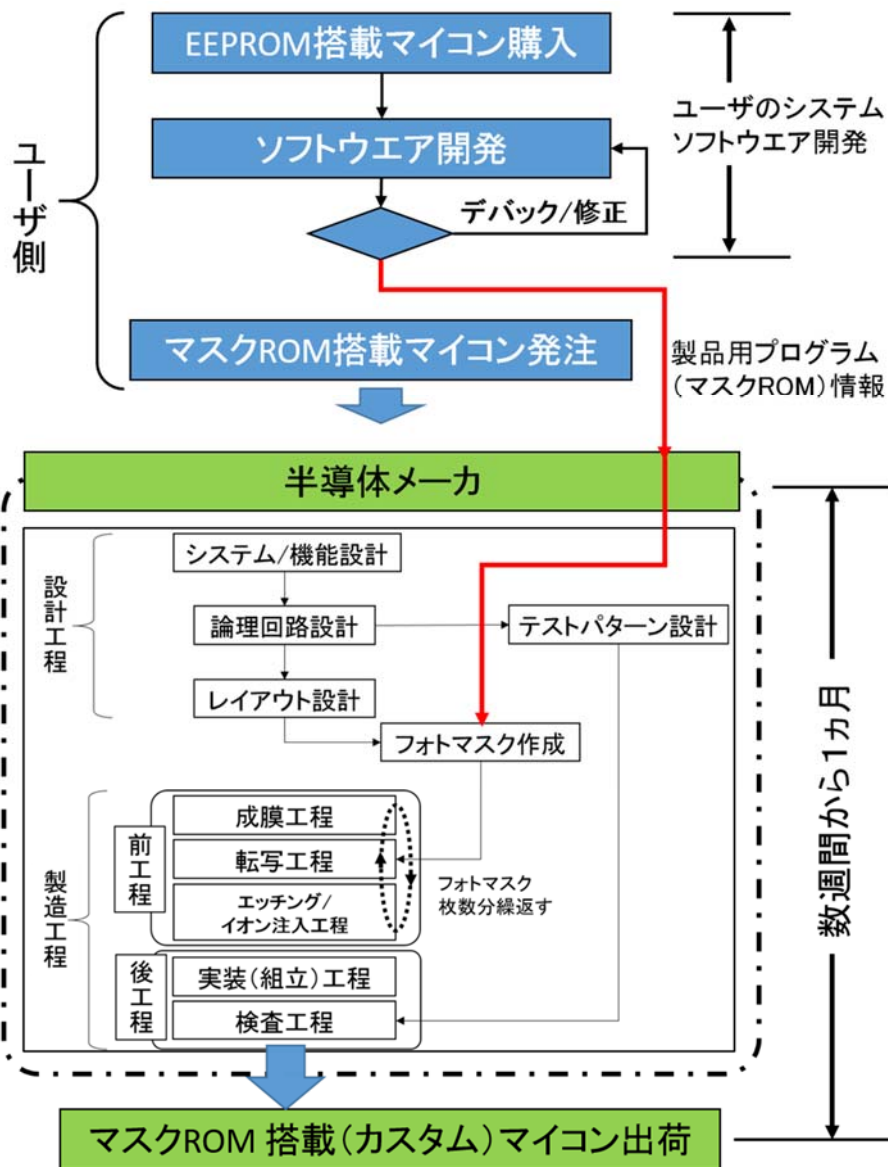


図 1-3 マスク ROM 搭載マイコン開発フロー

このような中、1980 年代後半になるとプログラムを一回だけ書き込み可能な One Time Programmable

ROM (OTPROM) を搭載したマイコン製品が各社で製品化された[1-5]。当時、エバチップ (評価用チップ) と呼ばれる Erasable Programmable Read Only Memory (EPROM) /Electrically Erasable Programmable Read Only Memory (EEPROM) オンチップマイコン等もあったが、コストが高くデバックや試作開発などの利用に限定されていた。ユーザが量産ベースで利用するマイコン製品はマスクプログラマブルなマスク ROM 搭載マイコンが主流であり、ユーザは自らプログラム開発完了後、半導体メーカーに委託して製造工程でマスク ROM に焼き込むことで初めてマイコン製品を利用することができた (図 1-3)。これはシングルチップマイコンの最大の弊害でもあった。シングルチップ化で小型化, 低コスト化が図られたが, このマスク ROM に焼き込む製造工程は量産ベースで Turn Around Time (TAT: 着工から完成までの期間) が, 数週間から一ヶ月程度の期間が必要であり, しばしば納期問題を引き起こすことがあった。また, ユーザのシステムノウハウの機密保持のため, ユーザプログラムの変更/バグ修正情報の連絡遅延やユーザシステムでの実機評価/検証時の環境構築等の問題が発生していた。しかし, OTPROM 搭載マイコンの出現により, 従来のマスク ROM 搭載マイコン製品に比べ TAT が大幅に改善された。

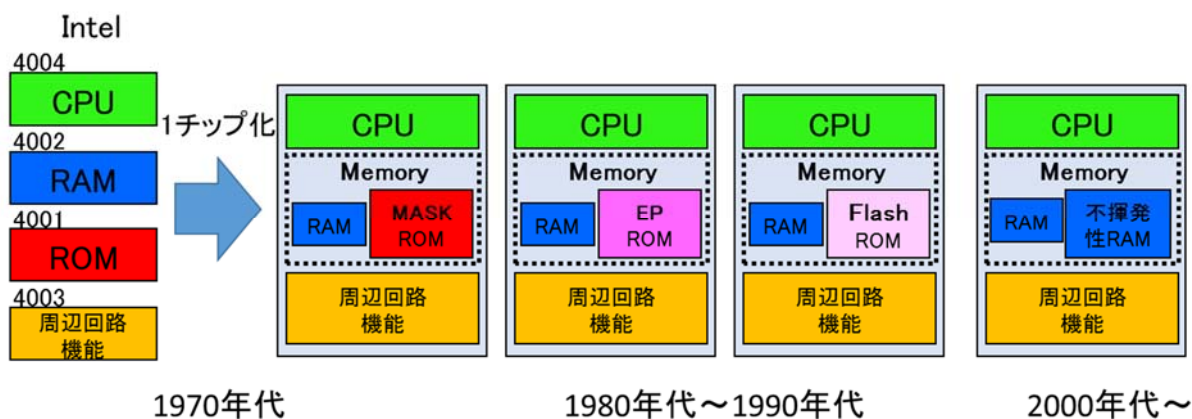


図 1-4 マイコン製品のプログラムメモリの変遷

ユーザはこの OTPROM 搭載マイコンにより, 半導体製造におけるマスク ROM 工程から解放され, 自らがプログラムをフィールドでプログラム開発・実装可能なフィールドプログラマブルなマイコンを手にすることとなる。さらにユーザは, システムノウハウを外部に出すことなく, システム出荷直前まで, プログラムの変更/バグ修正, 実機評価/検証も可能となった。その後, 1990 年前半にフラッシュ ROM を搭載したマイコン[1-6]が登場し, ユーザの利便性が格段に向上し, 本格的なフィールドプログラマブルなマイコン製品の時代に突入した (図 1-4)。

半導体メーカー側は, このフラッシュ ROM 搭載マイコンにより, ユーザのプログラムをマスク ROM へ焼き込む工程が省かれ, 未書き込み状態の ROM 製品をそのままユーザに出荷することができ, マイコン製品の製造から出荷までの TAT 短縮, コスト低減が可能となった。また, ユーザ側でも EP/EEPROM 搭載マイコン製品の書き込み専用装置も不要となり, フラッシュ ROM 搭載マイコンを標準品として購入しておけば, システムノウハウ (プログラム) を社外に出すことなく, システム製品出荷直前まで, 自らのプログラム開発やデバックが可能となった。さらにユーザはシステム製品出荷後でもプログラムの変更/修正ができるというメリットも享受できることとなった。図 1-2 に示したように, プログラムを実装するメ

モリの変遷とともにマイコン製品は発展し、フィールドプログラマブルなフラッシュ ROM 搭載マイコンが現れたことにより、半導体メーカ、ユーザ双方にメリットのあるビジネスモデルに大きく変化した。現在では、使い勝手の良いフィールドプログラマブルなマイコンとして、フラッシュ ROM 搭載マイコン製品が主流となり、幅広い分野で採用されている。2010 年代前後には Ferroelectric Random Access Memory (FeRAM/FRAM) 搭載マイコン [1-7][1-8] も登場し、限定的ではあるが利用が始まっている。将来的には Resistive Random Access Memory (ReRAM) や Magnetoresistive Random Access Memory (MRAM) などの不揮発性 RAM を搭載したものが登場することも期待され、さらに新しいマイコンの利用方法が生まれ、ビジネスモデルも変化すると予測される。

一方、半導体メーカのマイコン製品は、CPU コアをベースとした製品ファミリーが存在し、このファミリー毎に派生品種（プロダクトラインナップ）が準備される。これらマイコン製品ファミリーは、ユーザがターゲットシステムに最適な製品を選択できるように、CPU コアを基本とした動作周波数、内蔵メモリ容量で分類するとともに、カウンタ/タイマ、FIFO、通信インタフェース等の周辺回路を様々な組み合わせで準備している。さらにユーザの様々な要求に応えるため、基本的に同じアーキテクチャを有しながら、CPU コア、動作周波数、内蔵メモリ容量 (ROM/RAM) および周辺回路の組み合わせの違うマイコン製品を準備し、ユーザのカスタマイズ要求に対応してきた。これは、上述のインテルの MCS-4 と同様に、ユーザ向けにカスタマイズした製品を一定期間後、一般販売するといったビジネスモデルで対応を進めてきたが、長年にわたるビジネス継続により、各製品ファミリーにはこのような派生品種が多々存在する。多いものでは何千品種にも達し、I/O 数・パッケージ形状等すべてを加味すると数万品種にも達すると言われている。特に内蔵されているメモリと周辺回路に対する要求は種々多様であり、図らずも少量多品種ビジネスに陥っている。したがって、半導体メーカ側にとってファミリー内の派生品種数を削減することは極めて重要で、かつ早急に解決されるべき課題である。

マイコン市場として今後期待されている Internet of Things (IoT) 分野では、アプリケーション/サービスに直結する IoT 機器/エッジデバイス等への利用が期待されている。例えばセンサシステム対応のマイコン [1-9][1-10] は、IoT 等の新しい市場でも急速に拡大し、今後も様々な品種展開が必要になると考えられる。ビックデータを支えるこれら IoT 機器は、Cyber Physical System (CPS) 市場の大きなシェアを占めると予測されている [1-11][1-12]。この中でも、MCU とセンサをクラウドシステムに接続するセンサネットワークシステムは、サービスの種類によって多種多様なマイコン周辺回路が必要になると想定され、マイコン製品ファミリーの派生品種の数を減らすことは、将来的にも重要な課題である。

1.1.2 マイコンの市場動向

今後、半導体ビジネスは、少量多品種かつ、ロングテール市場になるとの予測もされている。これまでの半導体メーカでは、図 1-5 (a) のように低価格でも数量の多いコンシューマ分野、携帯電話等の市場を中心としていたが、人口問題、東日本大震災後のエネルギー問題、環境保護、経済発展地図の変化等社会・経済情勢の変化とともに市場も変化した。特に日本の半導体メーカは、同じ市場/地域でのビジネスの競争により弱体化し、吸収合併を繰り返してきた。また、近年は組み込み機器を中心とした市場が、IoT 市場に向かって大きく転換しはじめ、市場戦略も従来の大量消費市場狙いから、少量多品種を前提としたロングテール市場に対応する戦略が求められている。図 1-5 (b) に示すように、Programmable Logic Device (PLD) メーカは、ロングテール市場の高付加価値で、かつ数量の少ない幅広い市場からある程度

数量が見込まれる産業、自動車分野等への参入を目論む一方、従来の半導体メーカーは変動の激しいコンシューマ/携帯電話市場から脱却し、安定して一定規模の数量が見込まれる産業、自動車分野の市場確保に注力するとともに、新しい市場としてIoT分野に注目している。この分野では、サービスと直結したハードウェアが求められ、多種多様な半導体製品が求められると予想され、半導体メーカーとしても、IoTアプリケーション/サービスに対応するIoTプラットフォーム開発や新しいビジネスモデルの構築が必要となってくる。

現在、IoT分野ではセンサを利用したIoTデバイス/エッジデバイスの開発がすすめられ、Programmable System-on-Chip (PSoC)のようなセンサ部のインタフェースに必須のアナログ回路（オペアンプ/ADコンバータ等）機能とPLDを搭載するマイコンの製品化や利用が議論され、フィールドでカスタム対応可能なマイコンの市場要求が出始めてきている。しかし、ソフトウェア開発環境やハードウェア設計手法が複雑化し、ユーザの開発コスト・開発者スキル/負荷が増加する傾向にある。今後は、一つのハードウェア/プラットフォームとソフトウェアによるカスタマイズ可能なハードウェアの創生とその新しいビジネスモデルの構築が必要になると予想される。

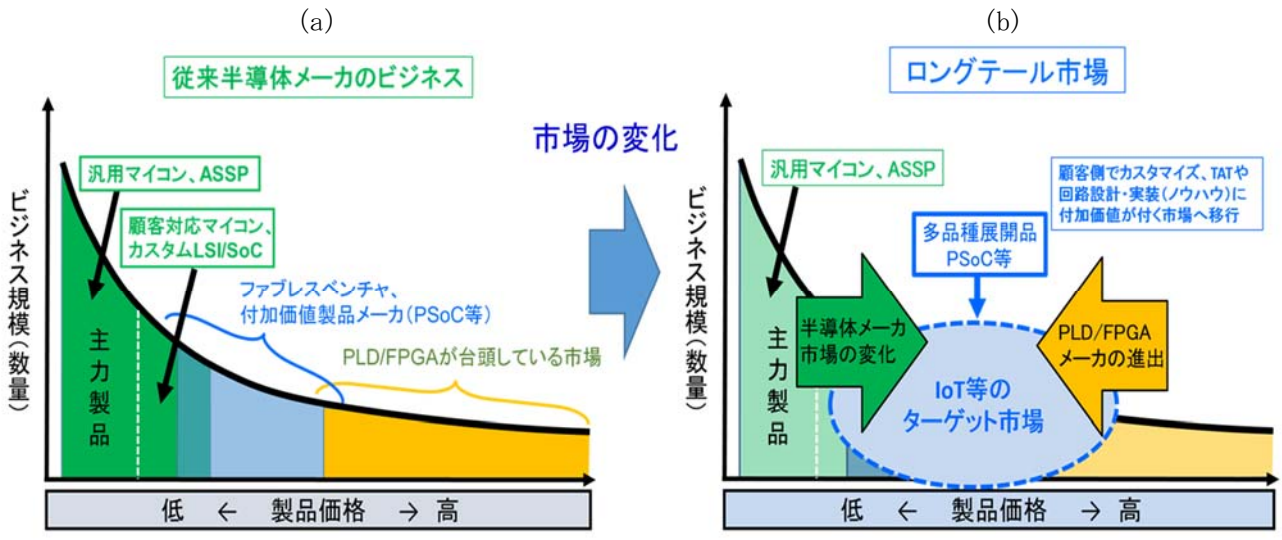


図 1-5 従来の半導体ビジネスとロングテール市場ビジネス

1.2 研究の目的

本研究の目的は、上述の背景のもと、周辺回路の違いによる派生品種を削減するため、1個のマイコンで様々な周辺回路が実装できるマイコン周辺回路に特化したプログラマブルロジックデバイスのアーキテクチャの提案である。また、対象とするマイコン周辺回路機能も合わせて提案し、その機能・性能を評価し、有用性を実証することを目的として行った。

具体的には、市場におけるマイコン製品の課題、プログラマブルロジックデバイスの技術課題を抽出し、マイコン搭載に適したプログラマブルロジックデバイスのコンセプトを明確にする(第2章)。次に、このコンセプトに基づき基本論理素子である Programmable Memory Unit (PMU) アーキテクチャをおよびマイコン向けに搭載する Field Programmable Sequencer and Memory (FPSM) アーキテクチャを提案する(第3章)。今回は、SystemCによるモデルベース開発手法を使って提案するアーキテクチャのモデル開

発と、そのシミュレーション評価および Field Programmable Gate Array (FPGA) による実装評価 (第 4 章) を行うとともに、実験チップの試作および評価 (第 5 章) を行った。さらに基本論理素子 PMU アーキテクチャを利用したパケットフィルタ応用研究について述べ (第 6 章)、最後に提案した基本論理素子 PMU アーキテクチャおよびマイコン向けプログラマブルロジックデバイス FPSM アーキテクチャに関するまとめ、実用性の考察と今後の課題 (第 7 章) について述べる。

1.3 本論文の構成

本論文は、上述の目的を達成するために行ったメモリをベースとしたマイコン周辺回路向けプログラマブルロジックデバイス FPSM アーキテクチャに関して研究成果をまとめたものである。本論文の構成と各研究の概略について以下に記述する。

第 1 章 序論

本研究に関するマイコンおよびプログラマブルロジックデバイスの歴史的、技術的背景、および本論文の研究目的・内容について述べる。

第 2 章 プログラマブルロジックデバイスのマイコン搭載への課題

市場におけるマイコン製品の課題、およびプログラマブルロジックデバイス技術動向と課題を述べる。さらにマイコン製品にプログラマブルロジックデバイスを搭載する上での課題を述べ、提案するプログラマブルロジックデバイスのコンセプトを提示する。

第 3 章 FPSM アーキテクチャ

本章では、FPSM アーキテクチャについて述べる。SystemC のモデルベース開発手法を使って、基本論理素子 PMU アーキテクチャを開発した。PMU は粗粒度のメモリを用い、マイクロ命令によって小規模なシーケンスプログラムが動作する。この PMU のシミュレーションモデルを作成し、マイクロ命令/アドレス制御の動作確認・評価を繰り返し、アーキテクチャの改良を行った。また、PMU を複数結線するためのスイッチボックス (SB) とアレイ構成およびマイコンに内蔵するための MCU インタフェースを組み合わせることで、内蔵メモリとして、かつマイコンの周辺回路としても利用可能な FPSM アーキテクチャについて述べる。

第 4 章 FPSM のシミュレーションモデルと FPGA 実装評価

PMU を複数組み合わせ構成したマイコン周辺回路をシミュレーションモデルに実装し、これらを実評価した結果を述べる。具体的には、PMU を用いたカウンタ/タイマ系、シフトレジスタ系、演算系の論理演算回路機能の動作確認、代表的なマイコン周辺回路機能 (FIFO, シリアル通信インタフェース, PWM) を PMU アレイ構成のモデル上に実装して動作確認を行うとともに、マイコン周辺回路を再構成可能なプログラマブルロジックデバイスとして利用可能なことを検証した。また、これらの中から 8 ビット精度の PWM を選び、FPGA ボード上に実装・評価した結果を述べる。

第5章 実験チップの試作および評価

今回提案した FPSM アーキテクチャを実証するために実験チップを設計・試作，および評価した結果を述べる．実験チップ上に各マイコン周辺回路機能を実装し，機能・動作を確認するとともに，設計した実験チップのハードウェア仕様，動作周波数，消費電力について評価を行った．また，限定的な実装回路ではあるが FPGA と FPSM 実験チップに実装した場合の実装面積および消費電力の比較を行った結果について述べる．

第6章 パケットフィルタ応用

基本論理素子 PMU アーキテクチャを利用したパケットフィルタ応用研究について述べる．一致/不一致検出回路を用いたパケット検索エンジンの一致回路部に PMU を利用したハッシュ回路を実装し，不一致検出回路と組み合わせて高スループット，かつ低消費電力なパケットフィルタ回路を提案した．この提案した検索エンジンの TEG チップの設計・試作，および評価した結果を述べる．

第7章 結論

本研究で得られたマイコン向けプログラマブルロジックデバイス FPSM アーキテクチャの研究についてまとめるとともに，製品化に向けた今後の課題と展開について述べる．

図 1-6 に本論文の構成について示す．

本研究成果が，今後のマイコン製品の新たな市場開拓，新たなビジネスモデルの創生に貢献することを期待する．

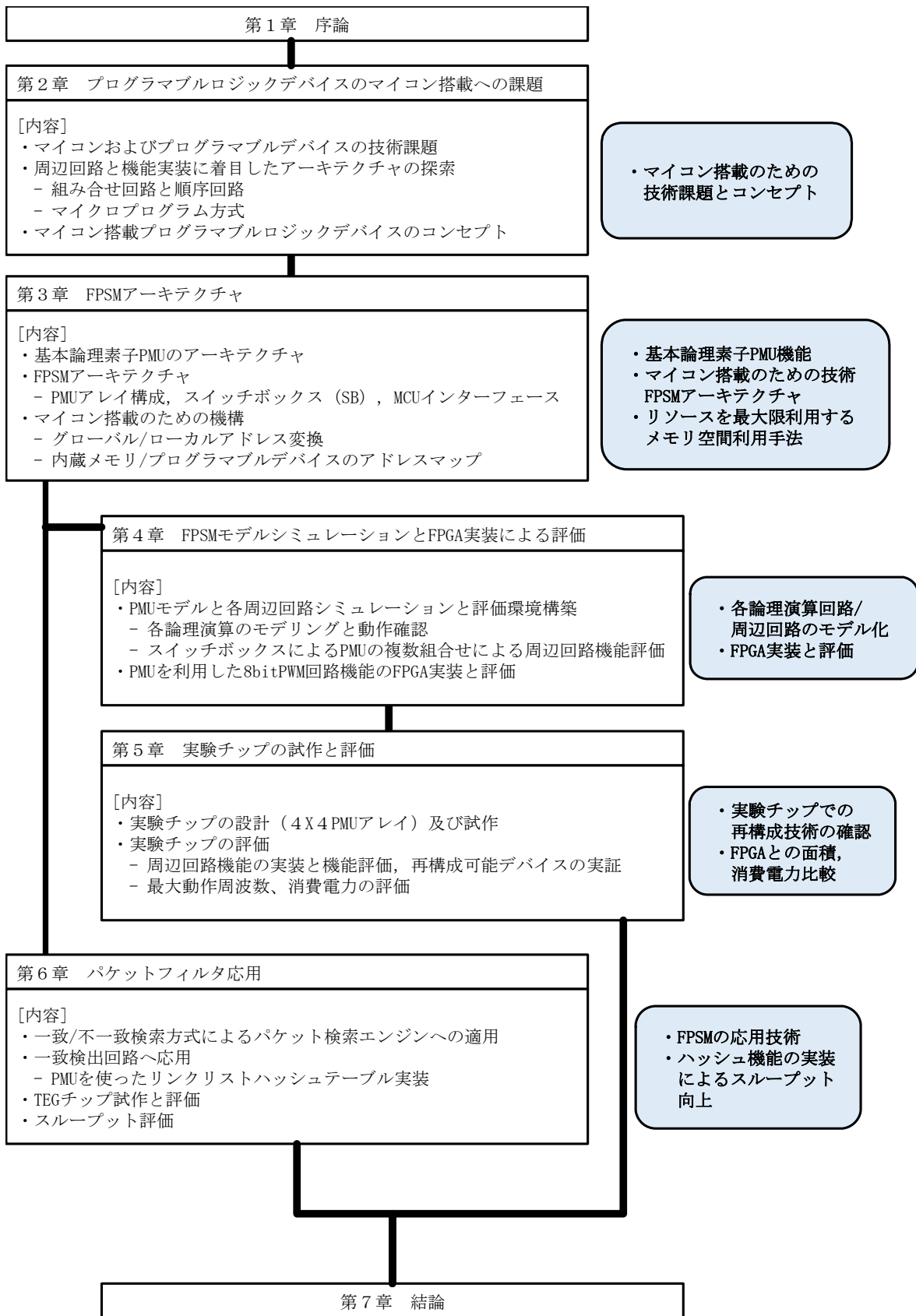


図 1-6 本論文の構成

参考文献

- [1-1] 電子情報通信学会“「知識ベース」6群コンピュータ-基礎理論とハードウェア 1編コンピュータの歴史 3章歴史的意義が大きいコンピュータ”
- [1-2] http://www.intel.com/Assets/PDF/DataSheet/4004_datasheet.pdf for “Data Sheet, 4004 SINGLE CHIP 4-BIT P-CHANNEL MICROPROCESSOR”
- [1-3] <http://www.intel.com/Assets/PDF/Manual/msc4.pdf> for “Manual, MCS-4 Four-bit PERALLEL MICROCOMPUTER SET”
- [1-4] <http://www.intel.co.jp/content/dam/www/public/ijkk/jp/ja/documents/corporate-information/history-intel-japan-2015ver1.pdf> for “インテルの歩み”
- [1-5] 佐藤恒夫, 新井保, “不揮発性メモリ内蔵マイクロコンピュータファミリ,” 日立評論 VOL.69, No.7, pp. 39-42, Jul.1987.
- [1-6] 館崎順一, 浅上浩明, 渡辺照一, “フラッシュメモリ内蔵マイコンの特徴と応用,” 日立評論 VOL.80, No.11, pp. 37-40, Nov.1988.
- [1-7] <http://www.fujitsu.com> for “Data sheet, MB95R203A”
- [1-8] <http://www.tij.co.jp> for “Data sheet, MSP430FR599x, MSP430FR596x Mixed-Signal Microcontrollers”
- [1-9] E. D. Kyriakis-Bitaros, N. A. Stathopoulos, S. Pavlos, D. Goustouridis, and S. Chatzandroulis, “A reconfigurable multichannel capacitive sensor array interface,” IEEE Trans. Instrumentation and Measurement, vol. 60, no. 9, pp. 3214-3221, Sept. 2011.
- [1-10] I. Adly, H. F. Ragai, A. El-Hennawy, and K. A. Shehata, “Over-the-air programming of PSoC sensor interface in wireless sensor networks,” Proceedings, IEEE MELECON, pp. 997-1002, Apr. 2010.
- [1-11] 喜連川優, “情報爆発のこれまでとこれから”, 電子情報通信学会誌, Vol.94, No.8, 2011年8月.
- [1-12] https://www.jetro.go.jp/ext_images/_Reports/02/ebc69b7777fbb2ad/NY_report_201508.pdf, 八山 幸司, “米国におけるIoT (モノのインターネット) に関する取り組みの現状”

第2章 プログラマブルロジックデバイスのマイコン搭載への課題

2.1 緒言

本章では、前章で述べた課題をさらに掘り下げるとともに、プログラマブルロジックデバイス技術動向と実装方法の課題を述べる。

現在、マイコン製品は汎用品でありながらユーザの要求に対応した派生品の開発や、自社独自の CPU コアをベースとしたマイコン製品ファミリは内蔵メモリサイズ、周辺回路、I/O 数、パッケージ等の派生品展開が行われ、これまでの過去の製品を含む品種数は膨大なものとなっている。これらは顧客毎/製品毎にフォトマスク作成とその保管管理、製品の在庫/出荷管理などの管理コスト増加を招いている。半導体メーカー側にとってマイコン製品ファミリ内の派生品種数を削減することは、事業全体のコスト削減につながる解決すべき重要課題である。この解決手段としてプログラマブルロジックデバイスをマイコンに搭載すること提案する。これに伴い、マイコン製品にプログラマブルロジックデバイスを搭載する上での課題述べ、既存のマイコン製品のアーキテクチャに無理なく導入可能であることを前提に、各プログラマブルロジックデバイスの構造、実装手法を分析し、従来のマイコンアーキテクチャを考慮し、新しく提案するマイコン搭載プログラマブルロジックデバイスのコンセプトを提示する。

2.1.1 マイコンの現状と今後の課題

【ユーザ側の現状】

ユーザがシステム設計を行う場合、設計者はトップダウンでハードウェア/ソフトウェア含むシステムの機能分割を行い、仕様・コストに見合った部品レベルに落とし込む。さらに各部品の配置配線、周辺装置とのインタフェースならびにユーザインタフェース設計を行う。その後、出来上がった実装ボードが、ソフトウェア開発エンジニアに引き渡され、ソフトウェアの開発・実装、デバックが行われ、システム全体の評価テストが繰り返され、完了する。

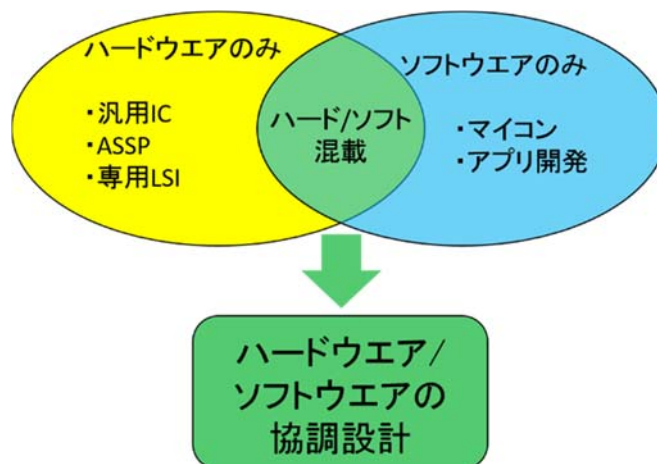


図 2-1 システム実現手段

システムを実現する手段としては、図 2-1 に示すように、ハードウェア、またはソフトウェアを中心としたアプローチが行われるが、最終的にはハードウェア・ソフトウェアの混載となる。近年はハードウェア部分をプラットフォーム化することで、アプリケーションソフトウェアの開発を中心としたシステム開発が進んでいる。典型的な例としては、パソコンやスマートフォンに代表されるコモディティ化したシステム構成である。組み込みシステムや今後期待されている IoT 機器なども、今後このようなプラットフォーム化が進みソフトウェアリッチなシステム開発が盛んになると予測されている。また、これらシステム実現の手段として、システムレベルでのハードウェア/ソフトウェアの協調設計手法の研究も進められている。

このようなシステム開発の過程において、エンジニアはできるだけ使いなれたマイコン部品を利用する人が多い。これはソフトウェアエンジニアの意向でもある。これまで利用経験のあるマイコンのソフトウェア資産が活用でき、開発期間も短縮可能となる。しかし、システム設計では、そのシステムの後継機種でありながら、機能の追加や性能向上が求められる場合、経験のあるマイコンが選択できない場合がある。上述のようにマイコン製品にはファミリーがあり、ファミリー内の派生品種をシステム仕様に合わせてピンポイントで選ぶ必要があり、往々にしてタイマあるいは Pulse Width Modulation (PWM) などの周辺機能が足りない場合が発生する。ユーザから「タイマや PWM が足りないから追加してほしい」と要求されても、開発費用や購入数量が見込めなければ半導体メーカー側としてはマイコン製品を一品種増やすことはできない。また、上述の Intel 4004 のようなビジネスモデルでない限り、半導体メーカーとしてはビジネスを辞退するしかない。このような場合、ユーザがソフトウェアで機能を追加することも可能であるが、組み込み機器でこのようなソフトウェア機能を追加すると、CPU に負荷が掛かり全体の機能/性能や割り込み等のタイミングに影響がでるため、できるだけ避ける必要がある。したがって、所望の周辺機能が必要な数をもつ上位機種あるいは他のファミリー製品を選ぶことになる。場合によっては他社の使用実績のないマイコンを選択せざるを得ない。経験値の高いエンジニアであれば対処可能でも、経験値の少ないエンジニアにとっては高いハードルとなる。最後にどうしても手に入らない場合は、PLD を追加して必要な機能を補填することになり、ハードウェア設計の追加作業が必要となる。このように、ユーザはマイコン製品選択時にマイコン製品のファミリー内の派生品種をピンポイントで選ぶ作業に多くの時間を費やすことになる。

【半導体メーカー側の現状】

半導体メーカーが準備するマイコン製品ファミリーには、CPU/動作周波数、内蔵メモリ等が同じ仕様であっても、周辺機能の種類/搭載数が異なる製品が多い。これは半導体メーカー側がマイコン製品を開発する際に、チップサイズの許容範囲内で、内蔵メモリや周辺回路に冗長を持たせ、多めに周辺回路を搭載するのが常套手段となっている。さらにこれらの配線をマスク工程、あるいはワイヤーボンディング等で、パッケージ外形と I/O ピンの配置仕様に合わせて配線を行い、周辺回路の組み合わせや I/O ピン配置を後工程で決めている。現状、マイコン製品の中でユーザが定義できるハードウェア部分は、内部の周辺機能をレジスタ設定により指定の I/O ピン切り換え、I/O ポートの入力/出力設定、プルアップ/プルダウンの切り替え等の入出力ポート部に限定されている。

また、以前からマイコン製品に FPGA 等のプログラマブルロジックデバイスを取り込む検討[2-1]がなされてきたが、特許 (USRE34383/R. H. Freeman), US464248/W. S. Carter [2-2]) および論理設計・実装設計

ツール等のハードウェア設計開発環境の課題もあり、なかなか製品化されなかった。しかし、2002 年になってマイコンに PLD を搭載した PSoC が製品化された [2-3]。この PSoC にはユーザモジュールと呼ばれる部分にアナログおよびデジタルの再構成部分を搭載している。デジタル再構成部は「デジタル・システム」と呼ばれ、Universal Digital Block (UDB) のアレイと、予め準備されたカウンタ/タイマ/PWM および通信インタフェースモジュールで構成され、利用する場合に内部結線で接続して利用する。さらにユーザ I/O モジュールを利用することで入出力ポートをフレキシブルに設定できるようになっている。この UDB は Complex Programmable Logic Device (CPLD) をベースに構成されており、ハードウェア/ソフトウェアエンジニア向けの専用の設計ツールを用いて実装される [2-4]。2012 年以降、IoT 機器の研究開発において各種センサ部との接続にアナログ再構成部分「アナログ・システム」が重宝され、センサネットワーク機器応用に多く検討されている。

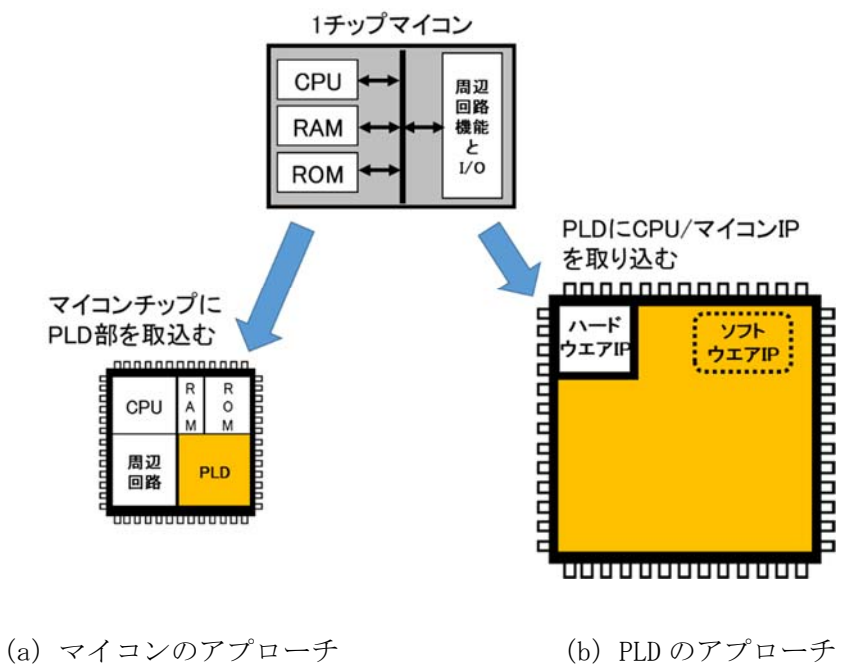


図 2-2 半導体メーカーのアプローチ

半導体メーカーでは図 2-2 に示すように、マイコンを主体とし、マイコン製品内に PLD を取込むアプローチ (a) と、PLD を主体とした PLD 内に CPU/マイコン Intellectual Property (IP) をソフトウェア/ハードウェア IP として取り込むアプローチ (b) がなされており、今後もこのようなアプローチが積極的に進められていくと予想される。さらに C 言語によるアルゴリズムレベルから Register Transfer Level (RTL) 設計に使用される Hardware Description Language (HDL) 言語に変換する環境等が提供され、設計効率・精度の向上を目指している。この PLD に CPU が実装される場合は、CPU コアの IP を提供するプロバイダー/ツールベンダの環境を利用してソフトウェアの開発を行うことになる。さらにトップダウン設計を目指したハードウェア/ソフトウェアのコ・デザインの研究も進められ、従来のハードウェア開発を優先したソフトウェア開発から、開発途中でもハードウェア・ソフトウェアの機能分割が変わっても対応可能な協調設計手法が研究されているが、ユーザ側の設計・実装スキルやコスト面で、まだハードルは高い。

2.1.2 プログラマブルロジックデバイス技術動向

図 2-3 にプログラマブルロジックデバイスとコンフィギュレーション用メモリの変遷を示す。プログラマブルロジックデバイスは 1970 年代から AND-OR アレイ構造にヒューズを用いた一回限り構成可能デバイスが市場に登場し、1980 年代半ばにはマイコンと同様に EPROM/EEPROM 等の不揮発性メモリの採用で繰り返し書換え可能なデバイスとして Simple Programmable Logic Device (SPLD) が、さらに Static Random Access Memory (SRAM) の Look Up Table (LUT) に Flip-Flop (FF) を加えた構造の基本論理素子を持つ FPGA などが登場した。1980 年代後半には SPLD をブロック拡張した CPLD、アンチヒューズ FPGA が製品化され、さらに 1990 年代半ばにはフラッシュ FPGA 等が登場する。市場では基本的に一回限り、または繰り返し再構成可能なデバイスとしては、Intel/CPLD と Xilinx/FPGA に代表され、現在も各メーカー主導のもと、デバイス開発と製品展開および設計ツールが展開されている。

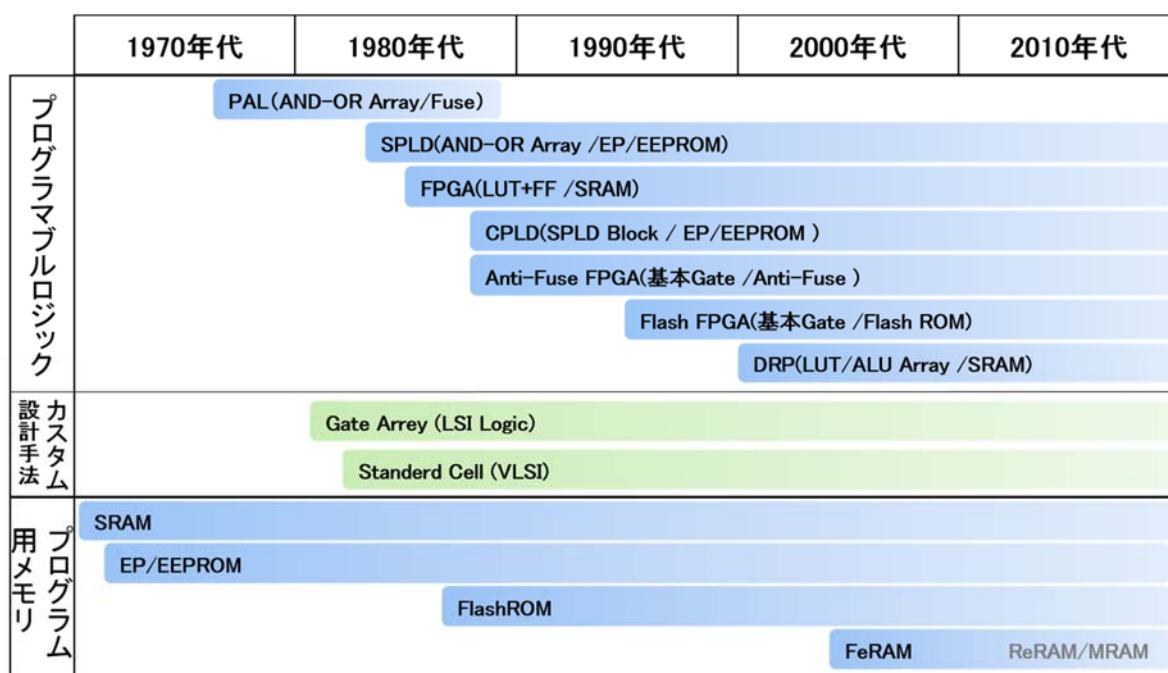


図 2-3 プログラマブルロジックデバイスとコンフィギュレーション用メモリの変遷

これらプログラマブルロジックデバイスは、システムあるいは LSI の一部を補う高速画像処理や通信インタフェース等の I/O デバイス応用等に多く利用されてきたが、2000 年代に入るとコンシューマ製品（デジタル家電/デジタル AV 機器）への採用も実現した。リソグラフィ技術、開発環境の改善に伴い、複数の CPU, DSP/コプロセッサ等々の実装が可能となり、ASIC/SoC と同様に利用された。最大の理由はコストである。半導体メーカーでカスタム ASIC を開発する場合、半導体プロセスの微細化が進み、フォトマスク等に費やされる開発費が膨大な費用になり、家電品一機種の開発では採算が取れない状況になってきたからである。また、2000 年前後から Dynamic Re-Configurable Processor (DRP) の研究が盛んになり [2-5]、2010 年前後に一部のユーザで映像機器、業務用印刷機等で採用されたが [2-6] [2-7]、デバイスコスト、消費電力、実装時のソフトウェアスキル、開発環境およびツール操作の複雑さも加え一般化していない。

将来的には、ヒューズ/アンチヒューズを除きこれら PLD はマイコンと同様にコンフィギュレーション用メモリの変遷とともに進化していくと予測される。特に FeRAM, MRAM 等の不揮発性 RAM は性能面、コスト面でも魅力的な材料であり、新たな製品展開のキーテクノロジーと考えられる。

現在、PLD は市場で一般化が進み、高集積度、低消費電力、量産効果によるコストダウンを実現し、製品サイクルの短いコンシューマ、携帯機器にも採用されるとともに、車載用にも検討され始めてきた。これらのプログラマブルロジックデバイスを記憶素子、基本論理素子で分類した結果を表 2-1 に示す [2-8]。

表 2-1 プログラマブルロジックデバイスの分類

アーキテクチャ	名称	記憶素子	基本論理素子	特徴
AND-ORアレイタイプ	PLA	ヒューズ	AND-ORアレイ	一回限り構成可能
	SPLD	EPROM	AND-ORアレイ+マクロセル	繰り返し再構成可能
	CPLD	EEPROM FlashROM	SPLDブロック	繰り返し再構成可能
細粒度タイプ	FPGA	SRAM	LUT+FF	繰り返し再構成可能
		アンチヒューズ	基本ゲート	一回限り構成可能
		FlashROM	基本ゲート	繰り返し再構成可能
ALUタイプ	DRP	SRAM	LUT、ALUなど各種	動的再構成可能

アーキテクチャとしては

- 1) AND-OR アレイ (プロダクトターム方式)
- 2) 細粒度タイプ (LUT 方式または基本ゲートの 2 種類)
- 3) Arithmetic Logic Unit (ALU) タイプ (ALU アレイ方式, その他)

に分類され、それぞれ利用されている記憶素子 (記憶方式) によって、一回限り、または繰り返し再構成可能かが決まる。これらは信頼性、使用される環境等の利用目的によってユーザが選択することになる。この中でも CPLD および FPGA は、大容量化を積極的に進め、最先端のリソグラフィ技術をドライブするキーデバイスの役割も果たしている。さらにハードウェア IP/ソフトウェア IP の取り込みを進め、複数の CPU/DSP や画像信号処理等の専用アクセラレータなど様々な IP の実装が可能となってきている。開発環境等も従来の HDL/RTL 設計環境に加え、さらに上位レベル設計可能な C 言語設計環境の研究開発が進められ、MATLAB などのアルゴリズム・シミュレーション環境とリンクしたアプローチも準備されるなど、ユーザの開発環境のバリエーションも充実してきている。

このように、PLD はプロトタイプング等の試作評価や生産数量の少ないシステムなどで、TAT やコスト面で ASIC に比べ有利とされ、開発環境の発達とともに積極的に利用されてきた。近年はカスタム/専用 LSI は、LSI 製造プロセスの微細化にともない LSI の開発・製造コストが飛躍的に上がり、PLD は ASIC の代用品として利用されるようになってきており、プログラマブルロジックデバイスの利用はますます増えてきている。

図 2-4 に各デバイスの位置づけを示す。カスタム IC/ASIC は専用設計されているため性能は高く、フォトマスクベースの高密度な実装が可能であるが、汎用性が低い。ASSP は特定分野・用途向けに限定的な範囲内で汎用性のある専用回路を搭載し、特定用途向けの性能も維持している。

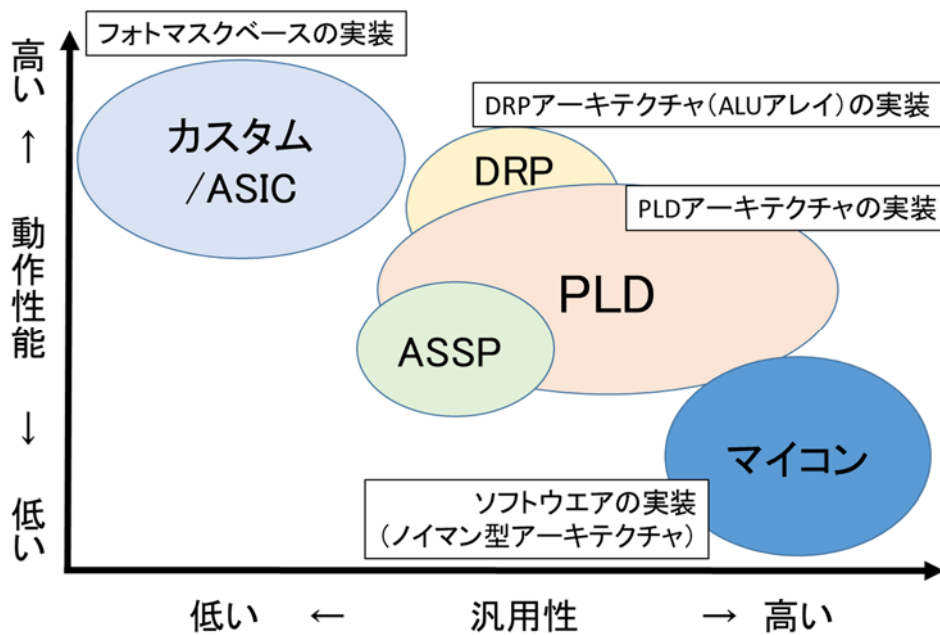


図 2-4 各デバイスの位置づけ

一方、PLDは汎用性が高いが、実装時の配置配線の遅延時間の最適化設計に性能が依存する。PLDは最新のプロセス技術で製造されているが、汎用性を高くするため基本論理素子およびこれらを自由に結線できる機構を実装しているため、製品のチップ面積は大きくなる。また、回路の実装効率もまだ低く、ユーザが実装する回路の集積密度は低い。さらに基本論理素子、配線機構が多く、配線が長くなると消費電力も増え、動作速度（遅延の発生）にも影響する。DRPのALUは、CPUと同じノイマン型アーキテクチャであるが、ALUアレイによる並列動作のため、性能は高いが消費電力が増加する。

マイコンはこれらデバイスの中では一番汎用性が高いが、性能は低い傾向にあった。近年はCPUのマルチコア化も進み性能も向上している。

表 2-2 に汎用性、性能、消費電力および機能実装上の難易度などの観点から各デバイスを評価した結果を示す。機能実装、性能、ハードウェア設計の自由度および消費電力設計面から考えるとカスタムIC/ASIC、またはASSPが望ましいが、開発費（コスト）、開発期間の点で厳しい。特にカスタムIC/ASICはユーザの開発費負担が重く、市場も少量多品種に移行しており容易に作れない状況にある。汎用性および開発費・開発期間の点で、標準ICであるマイコン、PLDが有利であるが、PLDでは実装手法、設計スキルによって実装効率に変化し、また単価の点で不利である。

性能や消費電力の面で、マイコンが優位となる。マイコンはソフトウェアによる柔軟性の高さゆえの有意差があるが、ハードウェアの自由度は全く無い。機能/集積密度については初めから所望の機能を集積するフルカスタムが一番有利である。マイコンは豊富な周辺回路機能とソフトウェアの活用をすることでユーザの要求を一定のレベルまで満たすことができるが、ハードウェアの柔軟性はないため、システムに依存したI/Oデバイスの追加を余儀なくされている。

表 2-2 各デバイスの比較

	カスタム /ASIC	ASSP	PLD (CPLD/FPGA)	マイコン	DRP (参考)
汎用性	×	△	○	◎	○
性能	◎	○	○	◎	◎
消費電力	◎	○	△	○	×
機能実装	◎	○	△	○	△
開発費 (コスト)	×	△	○	◎	○
	(多:◎, 小:×)	(多:○, 小:×)	(多:△, 小:◎)		(多:△, 小:◎)
開発期間	×	○	◎	◎	◎
設計の自由度, 性能・電力	◎	△	△	×	△
機能/集積密度	◎	○	×	○	×
プログラマビリティ	×	○	◎	○	◎
	(マスク)	(フィールド)	(フィールド)	(フィールド)	(フィールド)

2.1.3 マイコンとプログラマブルロジックデバイスの課題

マイコンとプログラマブルロジックデバイスの課題を以下にまとめる。

1) マイコンの課題

(1) ハードウェアの自由度が無い

OTPROM 搭載マイコン出現でソフトウェアの実装に関しては、フィールドプログラマブルになったが、ハードウェアはまだフィールドプログラマブルになっていない。

(2) 各社の膨大な種類のマイコン製品群から部品選択

依然として、ユーザは各半導体メーカのマイコン製品のファミリ/派生品の中から、所望の機能・性能を持つマイコン製品をピンポイントで選ぶ必要がある。

(3) チップサイズの許容範囲内で、内蔵メモリや周辺回路を多めに搭載するのが常套手段となっており、使われない周辺回路も多い。

(4) 部分的にハードウェアプログラマブル技術を採用

一部のマイコン製品ではプログラマブルな入出力ポートが搭載され、ユーザが I/O ポートを定義できる。また、PLD を搭載したマイコン製品 PSoC が 2002 年に製品化されたが、一般化はしていない。近年になって IoT 機器等のセンサとのインタフェース部に利用されはじめたが、アナログ再構成部分の評価が高い。

2) プログラマブルロジックデバイスの課題

(1) ハードウェア設計スキルが必要

PLD は、開発費/開発期間ともに従来のカスタム IC 開発に比べ有利であるが、各プログラマブルロジックアーキテクチャは、基本論理素子の構造、記憶素子および配線に依存した設計制約があり、さらに開発環境等の配置・配線技術等のハードウェア設計の知識とスキルが必要とな

る。さらに設計エンジニアのスキルのレベルによって、実装効率が変動（約 30～60%程度）する。

(2) 実装効率が悪い場合、使わない PLD リソースが多く存在する。

2.2 マイコン搭載プログラマブルロジックデバイスアーキテクチャの探索

上述の課題に対して、さらに掘り下げるため PLD および DRP のアーキテクチャ、実装手法といった点から分析を進め、これらプログラマブルロジックデバイスのマイコン搭載への適性を探るため、再度 PLD の定義を確認する、

- 1) プログラム可能な論理回路デバイスで構成されている
- 2) 標準品として購入可能である
- 3) ユーザが自ら（フィールドで）回路設計および実装が可能である
- 4) ハードウェアの仕様変更があった場合でも即座に設計変更の対応が可能である

以上の観点から、主要な PLD デバイスとして、CPLD、FPGA および DRP（ALU アレイ）を選択し、それぞれのアーキテクチャを分析する。

2.2.1 プログラマブルロジックデバイスの構造分析

図 2-5 に各プログラマブルロジックデバイスの特徴を示す。CPLD は、SPLD を基本ブロックとして、複数の SPLD をプログラマブルスイッチで結合された構成となっている。ロジック部およびスイッチ部分の遅延時間が一定になるように配置され、比較的設計が容易とされる。

FPGA は、ロジック部の Basic Logic Element (BLE) が、縦横に配置された配線部分の間に配置されるアイランド方式で構成されている。BLE を結合する配線の自由度が高い分、配線遅延時間は、結合する BLE の位置によって変化するため、注意が必要である。

DRP は、一般に ALU アレイで構成され、外側に隣接配置されたメモリのデータを ALU が演算し、その結果を次の隣接された ALU に伝達するといった具合に処理され、演算処理アルゴリズム、演算回数によって ALU 同士の結線や演算フローが動的に変わる [2-9], [2-10], [2-11]。また、ALU とメモリを混載した構成も提案されているが [2-12]、アプリケーションレベルの柔軟性は高いが、冗長性が高いためコスト高であるとともに、適用範囲も限られてくる。さらに LUT をカスケード接続する方式も報告されているが [2-13]、LUT を利用する場合、参照のために CPU アクセスが必要になり、CPU 性能/システム性能の劣化を引き起こす要因となる。

CPLD と FPGA の特徴からマイコン向けプログラマブルロジックデバイスの要件として下記が導き出される。

- 1) 物理的な配置配線の作業、特に RTL 設計時の遅延時間の最適化設計作業を最小化、または無くす事が望ましい
- 2) 細粒度の CPLD/FPGA の基本論理素子は、プログラマブルロジックデバイスとして利用していない時の利用方法は限定的な小規模メモリに限られる
- 3) メモリを利用する LUT が実装方法としては扱いやすいと考えられるが、現状の FPGA の基本論理素子である BLE では LUT のメモリサイズが小さく、実装はゲート回路レベルであり、かつ配置配線

を多用する。中粒度～粗粒度のメモリで、ゲート回路レベルより上位の機能を実装し、かつ配置配線を最小限度にすることが望ましい

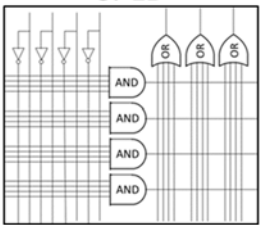
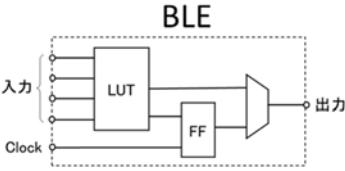
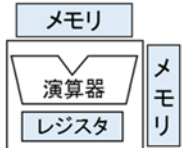
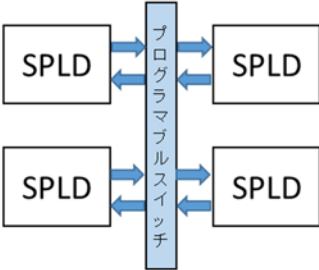
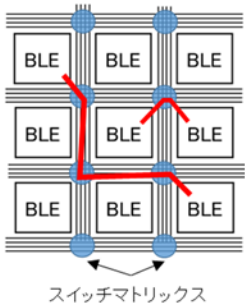

	CPLD	FPGA	DRP(参考)
組合せ論理 実現手法	プロダクト・ターム方式 (AND-ORアレイ)	LUT方式	ALUアレイ方式
粒度	細粒度	細粒度	粗粒度
記憶素子	不揮発性メモリ (EP/EEPROM, Flash)	SRAM, アンチヒューズ, Flash (SRAMの場合, コンフィギュレーション用ROMが必要)	SRAM (コンテキスト用、データ用)
設計レベル	RTL	RTL	演算命令レベル
集積度	小～大規模	中～大規模	小規模
基本論理 素子構造	<p>SPLD</p> 	<p>BLE</p> 	
配置構成			

図 2-5 各プログラマブルロジックデバイスの比較

以上のような状況とマイコン製品の特質および対象ユーザであるソフトウェアエンジニアを考慮し、プログラマブルロジックデバイスとして使用しない場合は、ある程度大きい容量のメモリとして利用できれば内蔵メモリとして活用できる可能性があり、マイコン製品・対象ユーザにとって親和性が高いと考える。

次に、LSI 開発時の情報処理のプロセスの観点から分析する。LSI の開発・実装をする場合、図 2-6 に示す情報処理のプロセスが必要となる。まず、①アルゴリズム設計を行い、②機能分割およびノーマン型コンピュータ処理を行うための逐次処理化を行う。次に論理演算のための③2 値化/真理値表を作成する。この真理値表から④論理式に表現するとともに、回路の動作・性能向上や部品点数の削減等の簡略化が行われ、⑤論理ゲート回路に置換（動作合成）、ハードウェアレベルであるトランジスタ回路に変換され、最終的に LSI 実装/配置配線に利用される。

これらのプロセスは、実装するデバイスのアーキテクチャに依存する。既存の LSI の実装では基本ゲ

ート回路で組み合わせ回路を表現し、CPLD ではこれを置換して AND-OR アレイにマッピングされる。FPGA では真理値表を用い論理式を簡略化し基本論理素子に準備されたルックアップテーブルのメモリ上にマッピングされ、組み合わせ回路を表現する。また、これ以外にも決定グラフを使い二分岐決定木での表現も可能である。

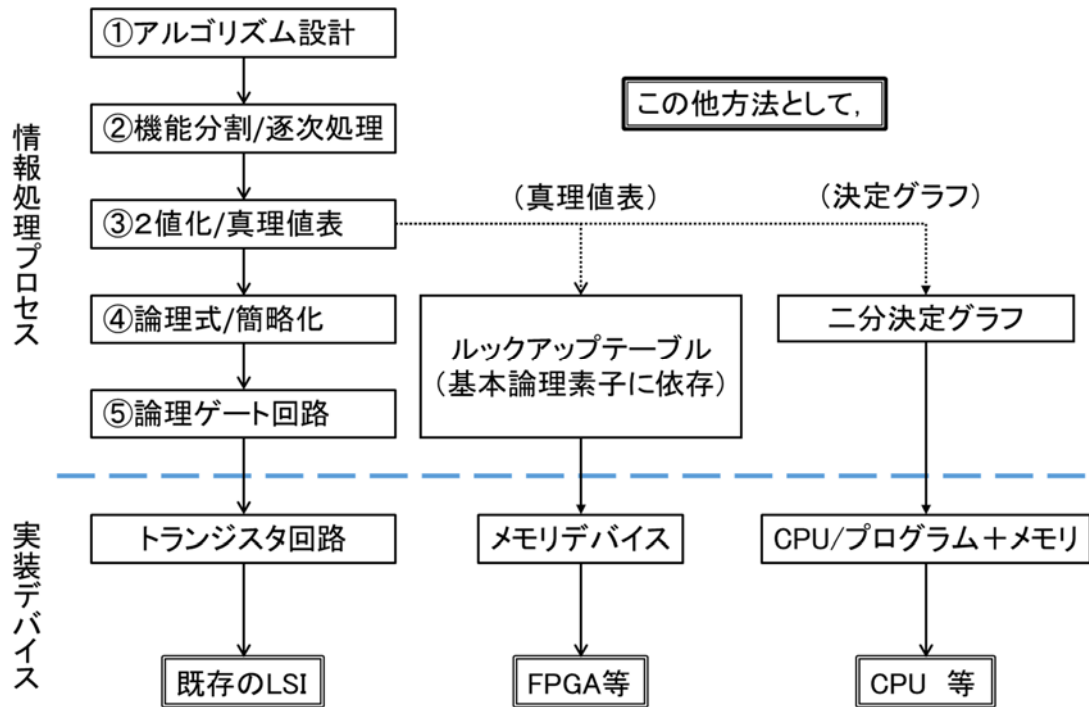


図 2-6 LSI 実装のための情報処理プロセス

各デバイスの構造および実装のための情報処理プロセスから、基本ゲート回路を使ったハードウェアの実装方法より、2 値化 (真理値表) を利用し、これらを LUT のメモリ上に実装して配置する、または CPU 上で動作するプログラムとメモリを利用した手法が有効と考えられる。

ここで、プログラマブルロジックデバイスで実現すべき組み合わせ回路と順序回路は、以下の定義となる。

- ・組み合わせ回路：ある時刻の出力信号が、現在の入力信号で一意的に決まる回路
NAND/NOR/ENOR やインバーター、マルチプレクサ等
- ・順序回路：ある時刻の出力信号が現在の入力信号と過去入力信号の影響を受けて出力が決定される回路 (記憶素子内蔵)
フリップフロップ、カウンタ、レジスタ等

これらは図 2-7 に示すように、FPGA の BLE を使って組み合わせ回路を実装できるが、規模によって複数の BLE を複雑な配置配線が必要となり、実装効率は良くない。また、順序回路実装する場合は、組み合わせ回路または FPGA 内にあるメモリブロックを利用して実装される。

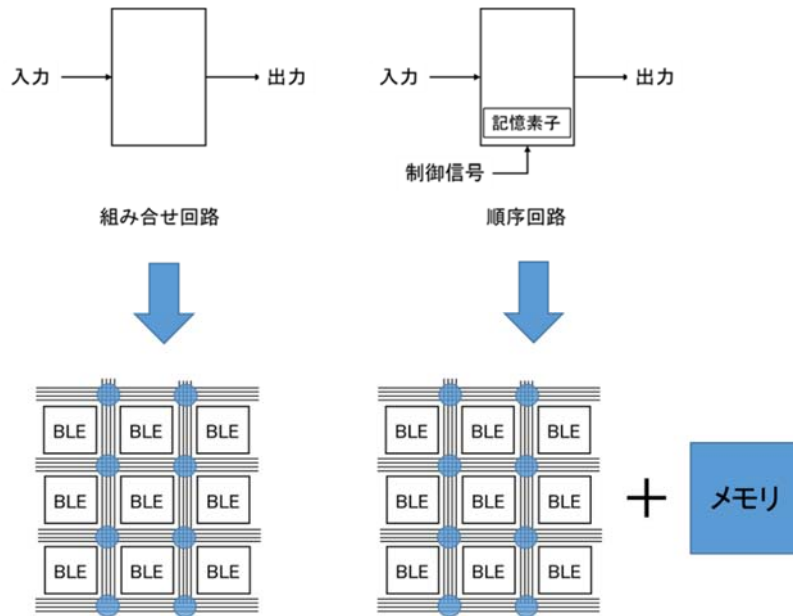


図 2-7 組み合わせ回路と順序回路の FPGA 実装

この順序回路にはムーアモデルとミーリモデルの2つがあり、それぞれの特徴を仕様に応じて実装する必要がある。特に、ムーアモデルは、簡単な回路構成で、出力にハザードが生じにくい特徴を持っており、このムーアモデルを前提に基本演算素子のアーキテクチャの探索を行った。

一般に CPU に用いられるシーケンス制御部を参考に、基本演算素子内の制御回路の制御方式の比較を行った。図 2-8 にムーアマシン（有限状態機械）の構成例を示す。このムーアマシンは、組み合わせ制御論理を実装した専用の組み合わせ回路と記憶素子であるレジスタで構成したものである。これらは、順序回路の仕様にしたがって、専用の組み合わせ回路で構成されるため、汎用性は低い。

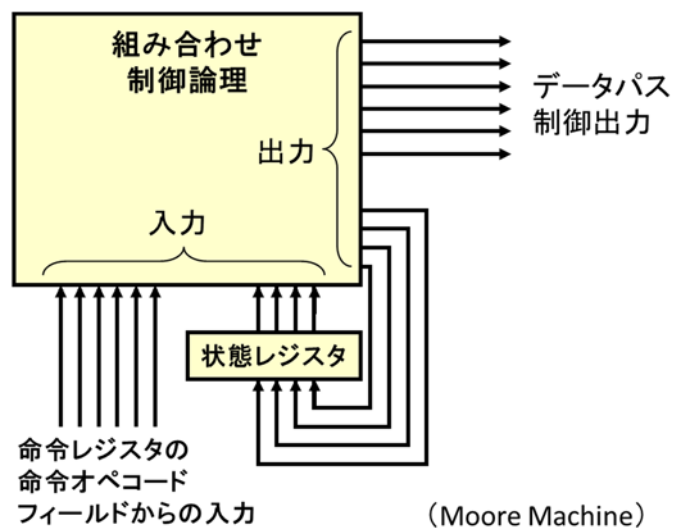


図 2-8 ムーアマシン（有限状態機械）の構成例

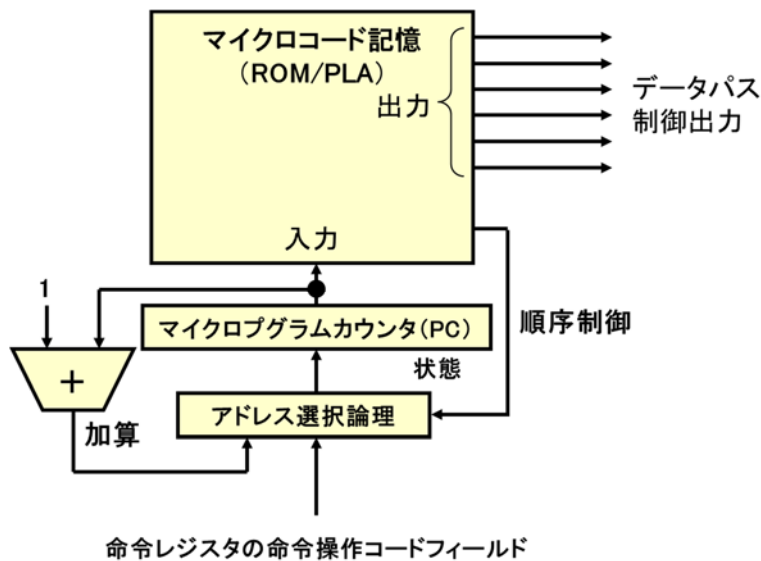


図 2-9 マイクロプログラム制御による順序回路の構成例

次に、図 2-9 にマイクロプログラム制御による順序制御回路の構成例を示す。このマイクロプログラム制御による順序制御回路は、CPU に採用されている順序回路と同じ構成となっている。マイクロプログラムを記憶する ROM/Programmable Logic Array (PLA) とアドレス制御回路部分で構成されている。制御回路部分の特徴はアドレス選択論理、Program Counter (PC) および PC のアドレス値のインクリメント用加算器で構成され、これらを使って入力するアドレスを制御することで順序回路を構成している。制御回路がシンプル、かつ極めて柔軟に対応できる汎用性の高い順序回路構成になっている。

ただし、ROM/PLA を利用して、上述の有限状態機械と同様に制御仕様を専用化する方法がとられている[2-14]。

2.3 マイコンアーキテクチャの比較

大型コンピュータやパソコン等では、CPU はプログラムメモリとデータメモリを同一メモリ上で取り扱うノイマン型アーキテクチャが一般的である。また、シングルチップマイコンでは、ノイマン型アーキテクチャとハーバードアーキテクチャの 2 種類が存在する (図 2-10)。汎用のマイコンアーキテクチャでは、バスの輻輳対策、コスト/消費電力等の理由からハーバードアーキテクチャが多く採用されている。これはメモリ利用の柔軟性より、比較的小型の組み込み機器に利用されることを前提に、プログラムメモリとデータメモリを使い分けることで、小型かつ動作周波数を低く抑えながら性能を維持し、低消費電力化、低コスト等のメリットを優先するためである。

プログラマブルロジックデバイスをマイコンに搭載する場合、このマイコンのアーキテクチャによって、実装手段は変わる。したがって、マイコンに搭載する場合、ノイマン型アーキテクチャとハーバードアーキテクチャの双方を考慮する必要があるとともに、マイコンとプログラマブルロジックデバイスの利用方法の親和性を高める必要がある。

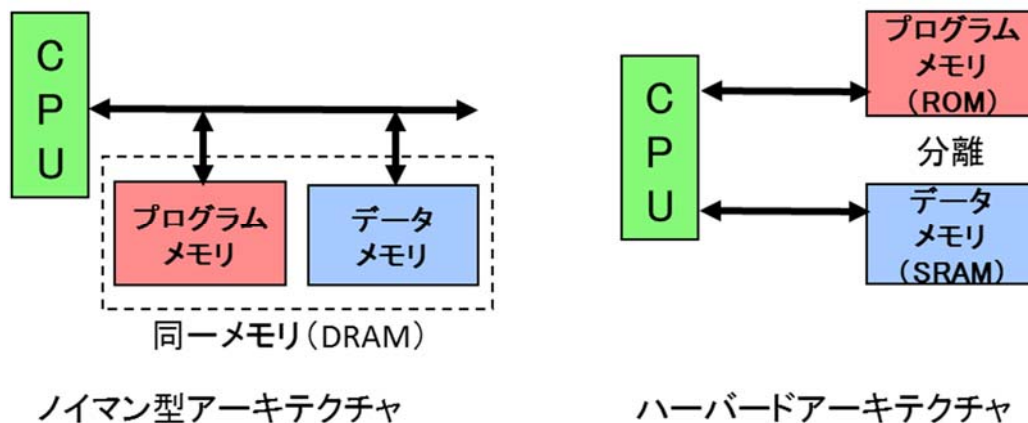


図 2-10 ノイマン型アーキテクチャとハーバードアーキテクチャ

2.4 結言

上述のマイコンおよびプログラマブルロジックデバイスの課題を踏まえて、マイコン製品の特質として、低コスト、使い勝手等、対象ユーザであるソフトウェアエンジニアにとってメリットのあるマイコン向けプログラマブルロジックデバイスのコンセプトとして、「CPU から通常の内蔵メモリとしてアクセス可能、かつマイコンの周辺回路としても利用可能」とし、さらに以下の要件を追加する。

- 1) プログラマブルロジックデバイスのリソースを最大限利用する
マイコンの周辺回路およびプログラマブルロジックデバイスには使われない部分があり、この使われない部分が有効活用できることが望ましい。
- 2) 周辺回路機能を実装する場合に RTL 設計スキルを必要としない
マイコンを活用するビジネスでは、各応用機器向けのプラットフォームが準備され、特に、組み込み機器系のユーザはこのプラットフォームをベースにシステム開発を進める。このようなシステム開発ではマイコンのソフトを開発するシステムインテグレータが主体となってシステム開発・評価を進める。このため、このようなプラットフォーム上でプログラマブルデバイスを搭載したマイコンをソフトウェアエンジニアが利用することを想定した場合、ハードウェア設計 (RTL 設計) スキルを必要としない実装設計が望ましい。
- 3) 従来のマイコン利用方法、CPU 性能に影響しない
メモリマップやアドレッシング、割り込み等々のマイコンのプログラムを開発する際に、特別なルールや制限を持たせないこと。すなわち、マイコンのソフトウェアエンジニア向けに、できるだけ従来のソフトウェア開発のルールに則った利用方法であることが望ましい。

以上の 3 点を追加し、マイコン向けプログラマブルロジックデバイスの開発コンセプトとした。

参考文献

- [2-1] 山崎尊永, “CQ 出版社技術解説 プログラマブル・ロジックを集積した SH マイコンのすべて - FPGA/PLD 市場に参入する日立製作所の取り組み

”<http://www.kumikomi.net/archives/2002/01/01shpld1.php?page=2>

- [2-2] 平成 18 年度特許出願技術動向調査報告書“リコンフィギャラブル論理回路,” (2007).
- [2-3] <http://www.cypress.com/> “Cypress Roadmap MCU Portfolio Q1 2017”
- [2-4] <http://www.cypress.com/> “PSoC® 5LP: CY8C52LP ファミリ データシート” (2016.10.26)
- [2-5] 末吉, 天野 : リコンフィギャラブルシステム, オーム社(2005)
- [2-6] “やわらかくなる LSI”日経エレクトロニクス, 2009 年 7 月 27 日号 (2009)
- [2-7] “動的再構成プロセッサ技術がついにデジカメに”日経エレクトロニクス, 2011 年 8 月 22 日号
- [2-8] 天野, et al. : FPGA の原理と構成, オーム社 (2016)
- [2-9] V. Baumgarte, G. Ehlers, F. May, A. Nüchel, M. Vorbach, and M. Weinhardt, “PACT XPP - A self-reconfigurable data processing architecture,” *The Journal of Supercomputing*, vol. 26, issue 2, pp. 167-184, Sept. 2003.
- [2-10] T. Sato, H. Watanabe, and K. Shiba, “Implementation of dynamically reconfigurable processor DAPDNA-2,” *Proceedings, IEEE VLSI-TSA International Symposium on VLSI Design, Automation and Test*, pp. 323-324, Apr. 2005.
- [2-11] H. Amano, S. Abe, Y. Hasegawa, K. Deguchi, and M. Suzuki, “Performance and cost analysis of time-multiplexed execution on the dynamically reconfigurable processor,” *Proceedings, IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 315-316, Apr. 2005.
- [2-12] G. Ansaloni, P. Bonzini, and L. Pozzi, “EGRA: A coarse grained reconfigurable architectural template,” *IEEE Trans. on VLSI Systems*, vol. 19, no. 6, pp. 1062-1074, Jun. 2011.
- [2-13] K. Nakamura, T. Sasao, M. Matsuura, K. Tanaka, K. Yoshizumi, H. Nakahara, and Y. Iguchi, “A memory-based programmable logic device using look-up table cascade with synchronous static random access memories,” *Japanese Journal of Applied Physics*, vol.45, no.4B, pp.3295-3300, Apr. 2006.
- [2-14] J.L.Hennessy, D.A.Patterson : *Computer Organization & Design*, Morgan Kaufmann Publishers (2005)

第3章 FPSM アーキテクチャ

3.1 緒言

本章では、マイコン向けプログラマブルデバイス FPSM アーキテクチャについて述べる。具体的には、プログラマブルロジックデバイスとして利用するための鍵となる基本論理素子 PMU アーキテクチャのモデル開発およびこの PMU で用いるマイクロ命令/アドレス制御の動作確認を行った結果を述べる。また、これら PMU を複数接続するために必要な配線機構であるスイッチボックス (Switch Box : SB) とこれらのアレイ構成、および CPU と接続するための MCU インタフェースについて述べ、これらの要素を組み合わせたマイコンに搭載可能な FPSM アーキテクチャモデルを開発した。これらは全て SystemC を用いたモデルベース開発手法を使って開発を行った。さらにマイコンに実装する場合の FPSM のメモリ空間の管理手法、プログラマブルロジックデバイスとしての利用方法について述べ、粗粒度メモリ群を内部メモリとして利用でき、かつプログラマブルロジックデバイスとしても利用可能な FPSM アーキテクチャについて記述する。

3.2 基本論理素子 PMU アーキテクチャ

前章のマイコンおよびプログラマブルデバイスのアーキテクチャの技術課題およびマイコン向けプログラマブルロジックデバイスのコンセプトを踏まえ、基本論理素子 PMU アーキテクチャを開発した。また、モデル開発にあたっては SystemC を用い、モデルベース開発手法を用いアーキテクチャモデルの開発を行った。

3.2.1 基本論理素子の検討

マイコンにはレジスタやカウンタ/タイマなどの機能部品が搭載され、多用される。また、PWM, FIFO および通信インタフェースなどの周辺回路もレジスタやカウンタ/タイマと連動して利用されている。これらの周辺回路機能は、CPU を使ってソフトウェアでも実装可能である。このように、メモリ、レジスタ、カウンタ/タイマおよび順序回路を利用し、シーケンスプログラムが実行されることで周辺回路機能は実装可能と考えられる。さらに CPU の負荷を掛けずに実行することができれば、従来にない新しいプログラマブルロジックデバイスの実装スキームの創生が可能である。従来の PLD では、細粒度の SPLD や BLE では最小単位の基本論理素子による上述のような実装の実現は厳しく、PLD 上に新たに回路設計・実装を行い実現するしかない。また、利用していない LUT をメモリとして利用することも可能ではあるが小規模に限られる。ある程度の容量のメモリを持ち、かつレジスタ、カウンタ/タイマと連動し、シーケンスプログラムが実行できる仕組みがあれば、マイコンの特質を生かす基本論理素子として最適と考える。

3.2.2 コンセプトと課題

ここで、前章で述べたマイコン向けプログラマブルロジックデバイスの開発コンセプトとその課題を再度整理する。

- ・コンセプト①：プログラマブルロジックデバイスのリソースを最大限利用する

PLDに使われている基本論理素子である AND-OR アレイと LUT を「有効活用」することを前提に比較し、実装されていない部分を他の機能に利用できると仮定した場合、

- (1) AND-OR アレイは、ハードウェアである論理ゲートの塊であり、利用されていない場合の再利用は難しい。
- (2) LUT は基本的にメモリそのものであり、プログラムまたはデータ用メモリとしての利用可能であるが、論理ゲートを表現するため 4~6 入力/1 出力の LUT では細粒度メモリで構成されており利用は難しい。したがって、LUT を内蔵メモリとして用いることが有効な手段と考えられるが、現状の LUT は細粒度メモリで構成されており難しい。内蔵メモリとして利用する場合は、ある程度粒度の大きい LUT が望ましい。

- ・コンセプト②：周辺回路機能を実装する場合に RTL 設計スキルを必要としない

プログラマブルロジックデバイスは、ハードウェア設計エンジニア向けであり、回路設計には RTL 設計手法とそのツールを操る知識とスキル、経験値が必要である。PLD 設計の経験のないソフトウェアエンジニアが利用している環境、またはこれに追加することでプログラマブルロジックデバイスの実装を可能とする手法が必要。

- ・コンセプト③：従来のマイコン利用方法、CPU 性能に影響しない

従来のプログラム開発のルール/スキルを踏襲することをで、ソフトウェア実装との親和性の高いプログラマブルロジックデバイスおよび実装方法であること。さらに、CPU の性能/ソフトウェアの実装に影響しないよう、ハードウェア同様に CPU から自立し、実装した機能が自律動作可能なことが望ましい。

また、新たに上述の「マイコンの特質を生かす基本論理素子」を鑑み、下記コンセプトを追加する。

- ・コンセプト④：カウンタ/タイマ機能を基本とする

マイコンの周辺回路にはカウンタ/タイマ機能を利用する回路が多い。

- (1) カウントダウタイマ/カウントアップタイマ
- (2) インターバルタイマ、ウォッチドックタイマ
- (3) インプット/アウトプットキャプチャ
- (4) コンペアマッチ (PWM)

など、これらを組み合わせて利用する。

以上、コンセプト①~④を考慮し、マイコン向けプログラマブルロジックデバイスでは、一つの基本論理素子で、カウンタ/タイマ機能を実装できること、また、このカウンタ/タイマ機能が実装可能な粒度のメモリを最小単位として利用できることを前提に基本論理素子モデル作成を行った。ここで、カウンタ/タイマ回路は順序回路で実現可能であり、検討する基本論理素子モデルは、順序回路が容易に実装可能なアーキテクチャとする。

3.2.3 カウンタ/タイマ機能を実装する基本論理素子モデルの検討

メモリで、かつ順序回路が実装可能なアーキテクチャの開発を行うため、SystemC を用いたアーキテクチャモデルの作成/評価を行った。

初めにワード長が 8 ビットのメモリモデルと 8 ビットカウンタの真理値表を作成し、メモリ入力アドレス値に対応したカウンタ出力データを準備し、実験を行った。スタート開始信号 (以下、Enable 信号)

を与えると、予め準備したスタートレジスタに記憶されたアドレスがメモリアドレスとして取り込まれ、メモリに記憶されたカウント値を出力する。入力されたアドレスに記憶されたデータが出力されるとともに、この出力値を次のアドレスとして利用するため帰還ループを設け、これを経由してメモリアクセスさせることで自律的にカウントアップを始める機構とした。

さらにカウントアップ時、指定のカウント値に達した場合の Carry Flag (CFLAG) 信号や、これらの信号を使ってカウントを終了させる制御回路が必要となる。そこで、これらの信号フラグ用のメモリと機構制御部を追加した初期のモデル (図 3-1) を作成し、SystemC でモデル化しシミュレーション環境構築と評価実験を行った。シミュレーション環境は、SystemC 2.1.v1、波形出力は、GTK Wave 3.0.19 (Windows XP SP2) の環境で行った。

モデル化時のメモリ構成は 8 ビット×256 ワードすなわち 2048 ビットの粗粒度メモリを前提としたが、フラグ信号 2 ビットが追加され、10 ビット×256 ワード構成のメモリが必要となった。出来るだけ特殊なメモリを利用しないため 2048 ビットの粗粒度メモリを 2 個準備し、一つは 8 ビットカウンタ用の真理値表を、もう一つにはフラグ信号用の 2 ビットを実装し評価を行った。

図 3-2 に SystemC でモデル化した初期の 8 ビットカウンタモデルのシミュレーション結果を示す。また、図 3-3 にそのシミュレーション波形を示す。

シミュレーション時の動作仕様は、

- ・動作仕様：8 ビットカウンタ上で、16 カウントし、カウント完了後停止する

とし、具体的には、Enable 信号により起動し、自律的にメモリ内でアクセスを繰り返す、16 カウント終了時に CFLAG を出力し、停止する一連の動作を確認した。

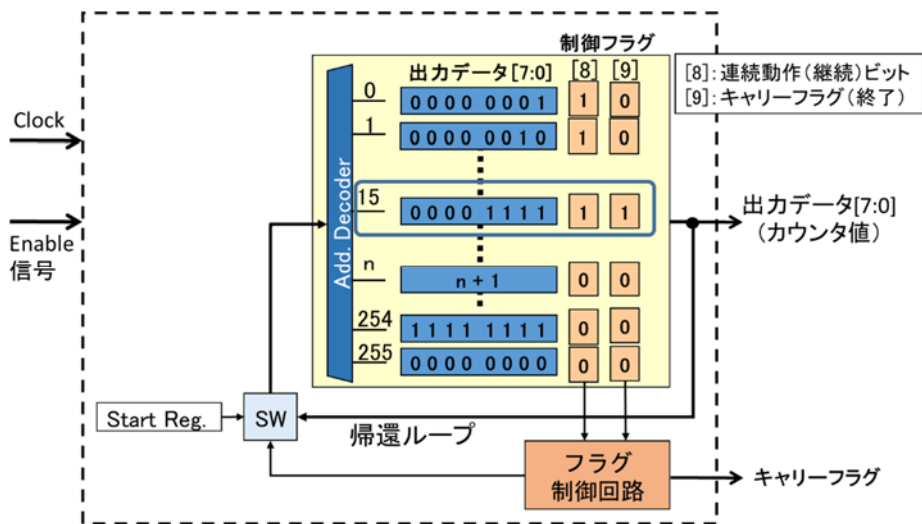


図 3-1 初期のメモリを使った 8 ビットカウンタモデル

図 3-2 のシミュレーション結果および図 3-3 のシミュレーション波形からも、カウンタ機能が問題なく動作していることを確認した。次に 8 ビットの繰り返しカウンタ、16 ビットカウンタおよび 8 ビット PWM の実装を行ったが、モデル機能を追加、変更する度にフラグ制御回路部分の追加変更の必要が出てき

た. 特に8ビットPWMを実現する場合, 複数の基本演算素子を結合し, 連係動作させる必要があること, それに伴うフラグ信号の追加およびこれらの信号を使って制御するための追加回路(論理演算回路)が必要であることが判明した. これは, 他の周辺回路を実装する場合も同様な事象が発生すると予測された. このため基本演算素子に新たな要件を追加した.

新たに追加する要件は,

- 1) カウンタ機能以外, 複数結線して利用する場合でも制御回路の変更無く使えること
- 2) 統一されたハードウェアで, かつプログラマブルな基本演算素子であること
- 3) 基本演算素子のメモリが従来のメモリデバイスとして利用できること
- 4) 制御回路は出来るだけ最小限度にとどめること

とし, 順序回路を, メモリを使って出来るだけ単純な仕組みで実現できる方式とした.

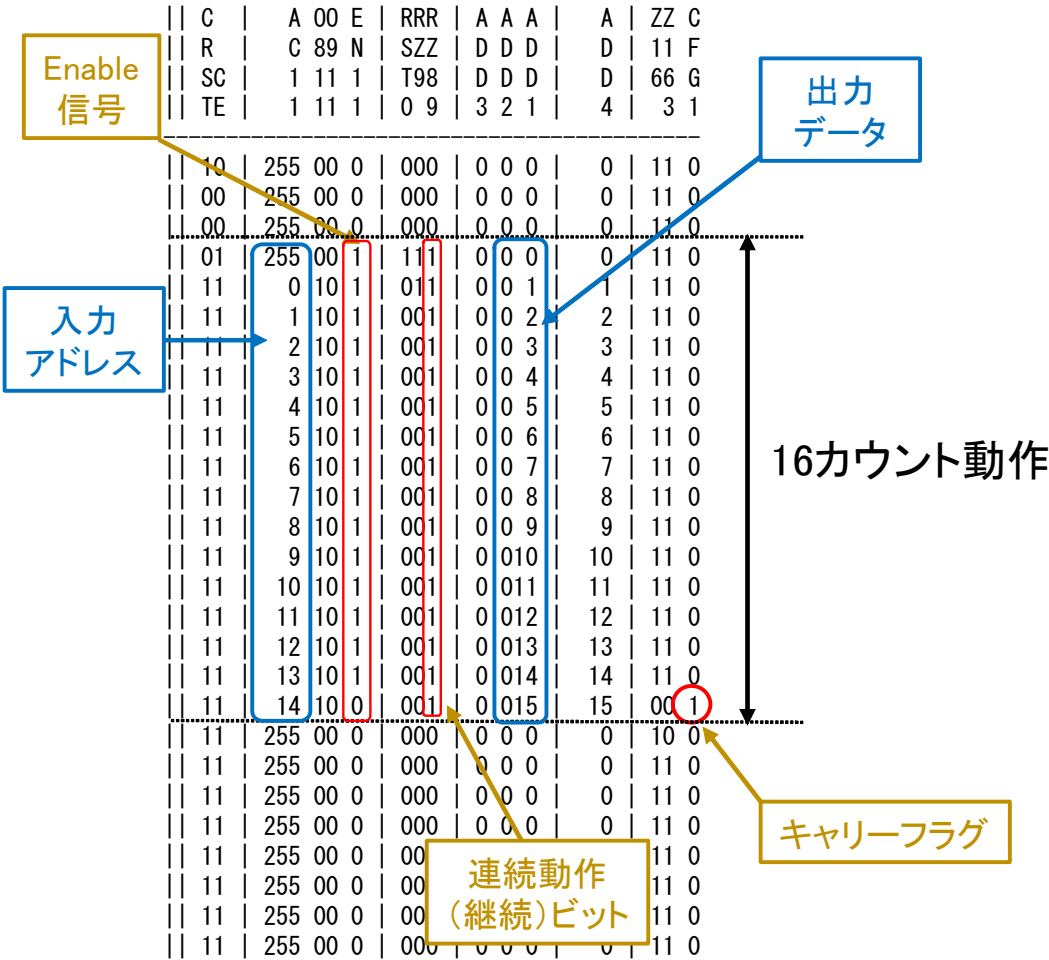


図 3-2 16 カウントのシミュレーション結果

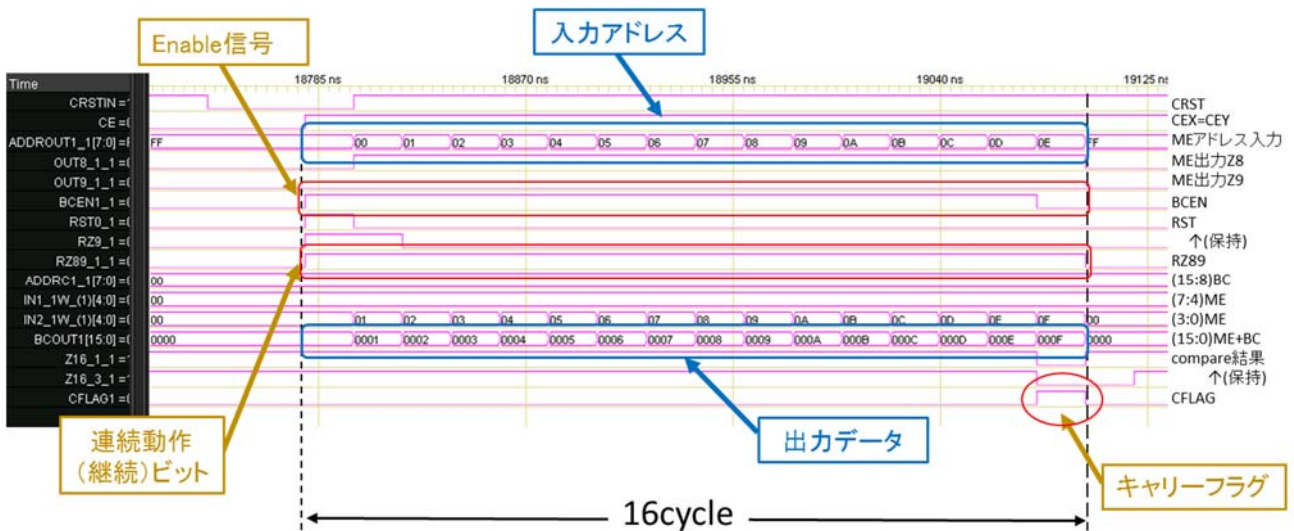


図 3-3 16 カウントのシミュレーション波形

ここで、前章で述べたマイクロプログラム（マイクロ命令）制御によるアドレス制御方式の導入を行った。アドレス制御方式は、構造的にもシンプルであり、汎用かつプログラマブルな順序回路を構成するには最適な手法である。そこで、初期の 8 ビットカウンタモデルにフラグ制御を拡張したマイクロ命令によるアドレス制御を導入し、制御回路部を一元化するとともに、マイコンに適したビット幅、ワード長を探索し、メモリ仕様の変更を行った。さらに、マイクロ命令を採用することで、シーケンシャルだけでなく、分岐（条件付/無条件）命令等の制御も可能とし、汎用性を向上することとした。図 3-4 にこの改良ポイントを示す。

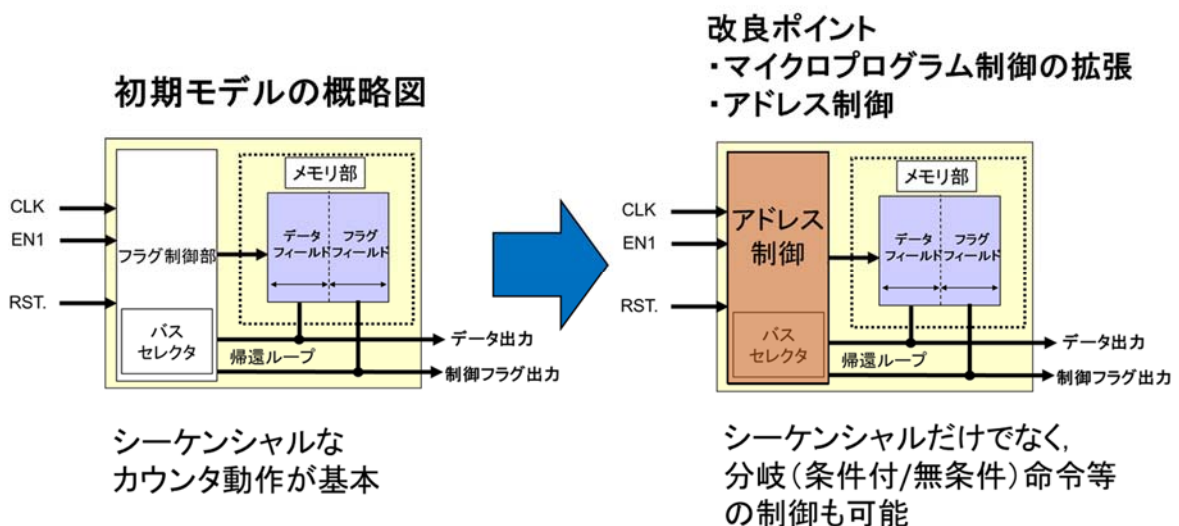


図 3-4 初期のメモリモデルからの改良ポイント

このモデルの改良は、アドレス制御方式の導入にあたり、フラグ信号のマイクロ命令の定義付けを同時

に進め、マイクロ命令の変更とシミュレーション条件の変更を繰り返し、モデル改良を進めた。また、この改良において、所望の機能を実現するために、メモリ容量・制御回路のゲート数の増加を許容し開発を進めた。シミュレーション環境は、上述と同様の SystemC 2.1.v1、波形出力は、GTK Wave 3.0.19 (Windows XP SP2) の環境で行った。以上により、図 3-5 に示す PMU アーキテクチャの基本モデルを開発した。

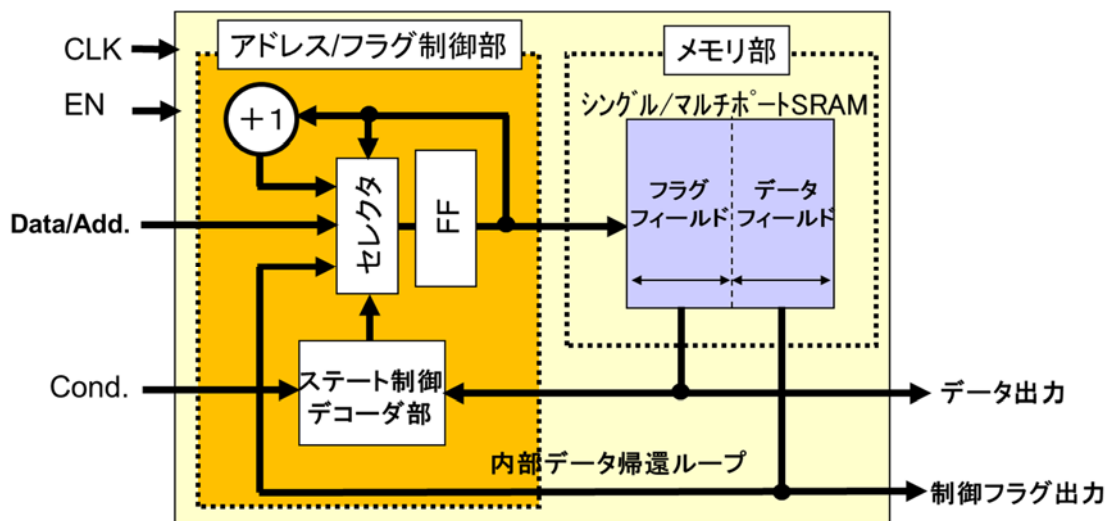


図 3-5 PMU アーキテクチャモデル

3.3 PMU のマイクロプログラム制御方式

この PMU アーキテクチャでは、CPU の PC の機能を実現するため、アドレス/フラグ制御部のセレクタ回路の後段にアドレスのラッチ回路 (FF) を挿入し、セレクタ回路と連動させることで入力するアドレスを制御する。この PMU アーキテクチャは有限オートマトンモデルになっており、PMU 単体でクロック、イネーブル信号/起動トリガ信号を用いて、自律動作が可能になっている。これは、アドレス/フラグ制御部の状態制御デコーダ (State Transition Decoder:STD) とメモリ部のデータ構造に工夫をするとともに、内部データ帰還ループを追加することで実現している。以下に PMU アーキテクチャモデルの特徴を列挙する。

- 1) 有限オートマトン (ムーアモデル) であり、入力に対して、一意に出力が決定する。プログラマブルなシーケンサ/順序回路が実装可能となる。
- 2) 同一アドレス上にデータとフラグが記憶される。データフィールドには現在 (t) 状態の出力、フラグフィールドには次 (t+1) 遷移状態が記憶される。あるアドレスがアクセスされると、データ出力バスから現在 (t) 状態の出力が出力される。同時にフラグ出力バスから次 (t+1) 遷移状態が出力され、アドレス/フラグ制御部に送られ、遷移状態を制御する。
- 3) アドレス/フラグ制御部はフラグによるマイクロ命令制御、複数の入力アドレスからセレクタ選択を実行する。このセレクタ選択より、無条件分岐が可能となる。
- 4) アドレス/フラグ制御部は上述 3) 項の機構により分岐命令が実行可能となり、さらに条件信号 (以下、Cond 信号) 等の情報を加えることにより、条件/無条件分岐を与えることが可能となる。

- 5) メモリ仕様は SRAM を想定し、16 ビット×256 ワード構成の 4096 ビットとした。プログラマブルロジックデバイスとして利用しない場合は、16 ビット×256 ワード構成の 4096 ビット内蔵メモリとして利用可能。プログラマブルロジックデバイスとして利用する場合は、下位 8 ビット×256 ワードをデータフィールド、上位 8 ビット×256 ワードをフラグフィールドとして利用する。データフィールドには実装する回路機能の真理値表を、フラグフィールドはその実装する回路機能実現するためのマイクロ命令が実装される。
- 6) 組み合わせ回路と順序回路をメモリ（真理値表）とアドレス制御で実装可能。カウンタ/タイマ、PWM、FIFO、調歩同期シリアル等の機能を実現可能。また、FPGA と同様に小規模の論理回路も真理値表で実現可能。ただし、メモリ構成が粗粒度のため、冗長性が高い。

3.3.1 PMU のマイクロ命令

PMU は、CPU からアクセス可能な通常の内蔵メモリ機能を維持するとともに、このメモリ部と密結合したマイクロ命令を用いたアドレス/フラグ制御部を採用することで、プログラマブルシーケンサとして動作する。一般にこのシーケンサ部は、次にどの命令を実行するのかを決める機構と命令アドレス生成回路を持っており、アドレス/フラグ制御部がこれにあたる。マイコンに実装される場合は、PC、プログラムメモリ、命令レジスタ、命令デコーダ、フラグレジスタ等で構成される。

通常の CPU の命令実行の動作ステップは、

ステップ①： 命令の読出し

PC 上に書き込まれたアドレスからそのアドレスが示すメモリ上の命令を読み出し命令レジスタにロードする。

ステップ②： 命令のデコード

フェッチされた命令が命令デコーダを介し、マイクロプログラム制御のフラグレジスタ経由、あるいは制御する機能・I/O モジュール等に制御信号が出力され、データの流れ（バス）を指定と同時に、PC は次の命令の読み出しを行う。

例： PC のアドレス値+1（アドレス値のインクリメント）

ステップ③： 命令の実行

指定された機能・I/O モジュール等で命令を実行する。

ステップ④： 次の命令準備

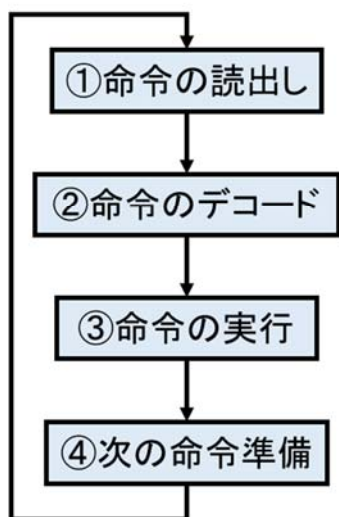
次の命令を呼び出すためのアドレスを準備する。

といった一連の動作を行う。また、もう一つ命令実行を制御する機能として分岐機能があげられる。現在のアドレス値から前方または後方のアドレスに分岐することで、実行すべき複数の処理の中から一つを選択した場合、その処理の先頭アドレスに分岐する、あるいは同じ処理を繰り返し実行する場合は末尾から先頭アドレスに分岐するといった分岐命令が必要となる。具体的には、条件分岐/無条件分岐命令があげられる。PMU アーキテクチャも上述と同様な動作を行う機構を持ち、これらはマイクロ命令によって制御される。PMU では、基本的に水平型のマイクロ命令の形式（図 3-6）を採用し、フラグフィールドの 8 ビットを利用して、ハードウェア制御、機能制御信号に用いられる。

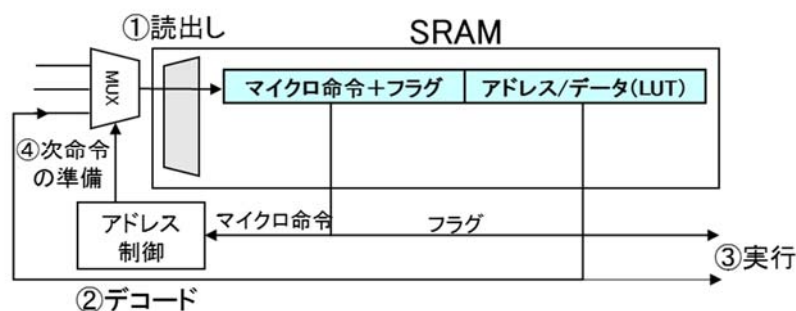
CPUでは逐次メモリから命令やアドレスを読み出す必要があり、アーキテクチャにも依存するが、命令実行まで数サイクル必要となる。PMUでは、メモリに密結合したセクタ回路とアドレス制御部があり、①命令の読出しが行われると、同時に②マイクロ命令・フラグデコードおよび③データフィールドのアドレス/データが出力（実行）される。カウンタ機能として利用する場合、「出力データ（カウント値）＝次アドレス」として利用されるため、③実行と④次の命令準備が同時に行われる。出力されたマイクロ命令はアドレス制御部に伝達され、次アドレスを入力するためにセクタ回路を制御し、④次の命令準備が行われる。フラグは出力と同時にハードワイヤードで結線された他の機能モジュール、I/Oモジュール等に伝達される。

ここで、当初のアドレス制御部分はセクタ回路を直接フラグで制御を行う機構としていたが、8ビットでは、将来機能拡張する場合に、ビット数が増える可能性が出てくるため、現状4ビットで制御していたセクタ制御信号を2ビットに圧縮するデコーダを追加した。

CPUの命令実行ステップ



PMUの水平型マイクロ命令実行ステップ



SRAMのアクセス時間、読み出しサイクル時間等SRAMの仕様に依存するが、基本的に1サイクルで①～④を実行する。

図 3-6 PMU の水平型マイクロプログラム制御方式

図 3-5 に示した PMU の基本アーキテクチャモデルの概要について述べる。PC 機能として、メモリのアドレスデコーダの前段に FF を挿入し、ここにラッチされたアドレス値が加算器（インクリメンタ）を経由して、次のアドレス生成を行う。さらにフラグレジスタ、命令レジスタおよび命令デコーダ機能をアドレス/フラグ制御部に一体化した。

メモリ部はワード長が 16 ビットの 256 ワード、すなわち 4K ビットの SRAM を採用し、デフォルト状態（メモリモード）では通常の内蔵 SRAM をワークメモリとして利用できる設定になっている。これはマイコン上での利用を想定し、プログラマブルロジックデバイスの実装（コンフィギュレーション）を行う場合、ソフトウェアエンジニアが実装するプログラマブルロジックデバイスのコンテキストを通常の内蔵 RAM に書き込むだけで良く、その後ロジックモードに切り替えるだけで、所望の周辺回路機能が利用できることを想定している。

PMU のマイクロプログラム制御はハードウェアで構成されたアドレス/フラグ制御部とメモリの上位 8 ビットに実装されフラグ信号によるマイクロ命令と下位 8 ビットのデータ/アドレス情報を時間的に連動させることでソフトウェア動作を実現している。

このメモリはフラグフィールドである上位 8 ビット、データフィールドである下位 8 ビットの出力が分割されており、上位 8 ビットはアドレス/フラグ制御部の STD に入力されるパスと後段に出力されるパスを持つ。下位 8 ビットはセクタ回路の入力につながるパスと後段に出力されるパスを持つ。ここで、上位 8 ビットはアドレス/フラグ制御部の STD に入力されるパスを経由してフラグ信号とマイクロ命令を伝達する。これらのフラグ・マイクロ命令は STD に準備された専用デコーダにより解読され、PMU 内の制御、次段の PMU、または周辺回路を直接制御する。さらに、この STD に Cond 信号等を与えることで、分岐命令時の条件を与えることができ、条件分岐命令が実行される。このように STD はこの条件信号とフラグ信号をデコードし、入力信号のセクタ選択部を制御する。セクタへの入力には、①内部データ帰還ループ、②外部直接入力、③加算器 (+1 : インクリメンタ) および④アドレスホールドの 4 つの入力から一つを選択し、メモリに入力するアドレスを選択することができる。

以上のように、PMU のシーケンスプログラムはアドレス/フラグ制御部のセクタによるアドレス選択とメモリの上位 8 ビットに実装されたフラグ・マイクロ命令および下位 8 ビットのデータ/アドレス情報が連動する事で実行される。

一般にプログラム内の分岐表現は、

```
Go to < ラベル名 > : 分岐命令,
if < 条件 > then go to < ラベル名 > : 条件付分岐命令
```

で表現され、CPU によって実行される。PMU では、図 3-7 に示すようなデータ構造で特定のメモリアドレス上に実装する。PMU に分岐命令を実装する場合は、フラグフィールドには、マイクロ命令定義した 8 ビットの分岐命令と制御フラグで条件分岐・無条件分岐等を実装し、データフィールドには分岐する 8 ビットのアドレスを指定することで、上記ソフトウェアと同等の表現となる。

	フラグフィールド(8bit)	データフィールド(8bit)	備 考
	内蔵メモリ	データ	
シーケンサ/ 組み合わせ回路	マイクロ命令+フラグ	アドレス/データ(LUT)	基本
	分岐命令+フラグ	Branch address	分岐命令の例

図 3-7 PMU のデータ構造

PMU でシーケンス動作させるプログラムを実装する場合、ソフトウェアのようなシンタックスルールは

無く、メモリがアクセスされた時点での、現在のデータフィールドの出力データとフラグフィールドの実行命令/制御信号を実装する。メモリのアドレスとこのデータ構造がリンクする事により、1 命令ステップ/1 サイクルのシーケンスを実行する。PMU が 256 ワードの場合は、256 ステップのシーケンスが 256 サイクルで実行される。

次に、PMU 上に 3 ビットカウンタを実装する場合を例に、マイクロ命令/フラグの定義方法、真理値表の実装およびその動作について述べる。

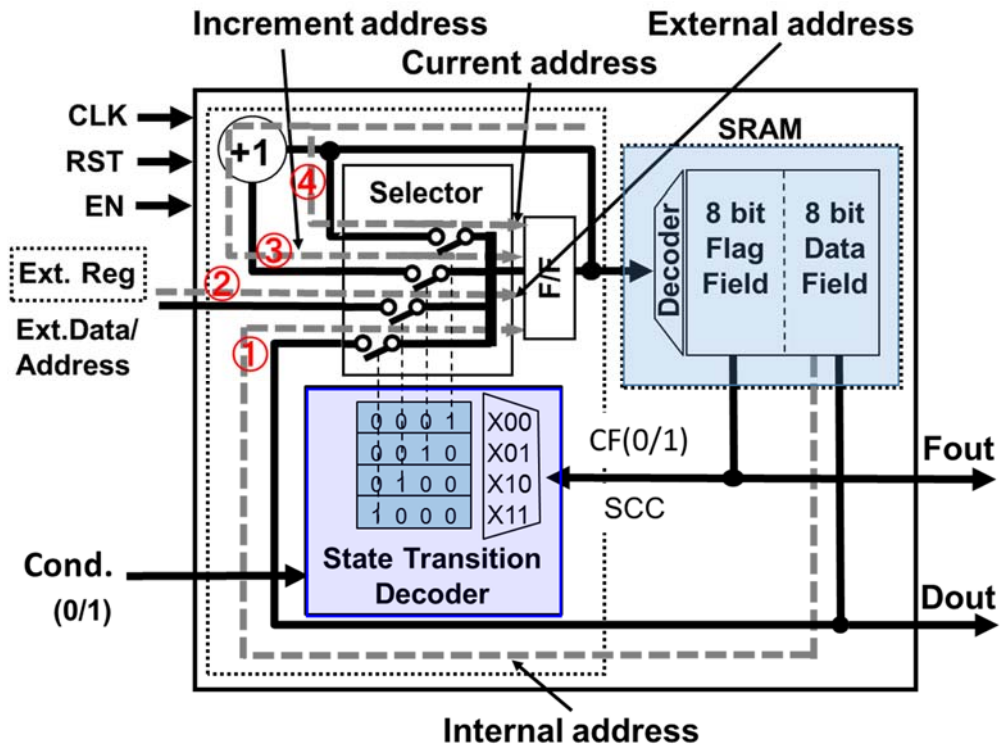


図 3-8 ステート制御デコーダ (STD) によるセレクタ制御

ここで前提条件として、

- ・実装するカウンタは 3 ビットダウンカウンタとする
- ・アドレス値は外部レジスタから与える任意の値からダウンカウント開始し、ダウンカウント終了時に CFLAG を出力するとともに、次の任意のアドレス入力を外部 (Ext. Reg) から入力できるように設定する。

以上の要件で、必要なフラグを設定する。ここでは、カウントダウン終了 (出力データ値: “000”) 時に CFLAG の出力が必要になるため、

CFLAG: 1bit[0]

を定義する。

次に、PMU のセレクタの選択信号 (SSC) は当初 4 ビットで制御していたが、圧縮するためデコーダを追加し 2 ビット構成とした。図 3-8 に示すように、セレクタの切り替えスイッチ 4 点 (①内部データ帰還ループ、②外部入力、③インクリメントおよび④ホールド) に対してセレクタ制御命令を 2 ビット、

SCC (Selector Control Code) : 2bit[1:0]

で定義する (表 3-1).

表 3-1 セレクタ部の入力選択フラグの定義

SCC	内 容
00	ホールド (FFにラッチされたアドレス保持し入力する. デバックなどに使用)
01	インクリメンタ (FFにラッチされたアドレス+1で次アドレスを生成)
10	外部入力 (分岐命令や外部演算結果の取り込み)
11	内部データ帰還ループ (出力データをアドレスとして使用)

以上により, 図 3-9 に示す 3 ビットダウンカウンタの真理値表, マイクロ命令/フラグの設定を行う.

Input Address	Output Data			マイクロコードでの表現
	Flag/code	LUT Data		
Bin(Dec)	CF	SCC	Next Add	
000(0)	0	11 ①	[Ext. Reg]	A000 CF=0; go to [reg];
001(1)	1	10 ②	000(0)	A001 CF=1; go to A000;
010(2)	0	11 ①	001(1)	A010 CF=0; go to A001;
011(3)	0	11 ①	010(2)	A011 CF=0; go to A010;
100(4)	0	11 ①	011(3)	A100 CF=0; go to A011;
101(5)	0	11 ①	100(4)	A101 CF=0; go to A100;
110(6)	0	11 ①	101(5)	A110 CF=0; go to A101;
111(7)	0	11 ①	110(6)	A111 CF=0; go to A110;

① : 内部データ帰還ループ, ② : 外部入力 [Ext. Reg]

図 3-9 3 ビットダウンカウンタの実装データとアドレス選択例

リセット後, PMU にクロックが供給され, メモリ上に 3 ビットのフラグと 3 ビットの次アドレスデータをそれぞれのフィールドにロードする. 任意のタイミングで外部からイネーブル信号を供給する事によりカウントダウンを始める. この時, ロジックモードではメモリのライトは保護され, 書き込みは出来ない状態となっている. リセット時, FF は “000” となり, メモリアドレスが “000” の状態が, 初期状態となる. ここでは, アドレス “000” では②外部入力状態に設定され, この外部レジスタから任意のアドレスが入力され, これを初期値としてカウントダウンを開始する. 外部レジスタからの入力アドレスを “111” として説明する. 入力アドレス “111” が入力すると, フラグフィールドとデータフィールドが読み出され, 命令の呼び出しが行われる. 出力データ “110” がメモリから出力されると, 命令の実行と次の命令が呼び出され, 同時に SCC “11”, すなわち①内部データ帰還ループ設定命令が STD に伝達され,

初期値で設定されていた②外部入力状態から①内部データ帰還ループにセレクタを切り替える。これにより先ほどメモリから出力されたデータ“110”が①内部データ帰還ループを經由して、今度は次アドレスとしてメモリに入力される。以降、同様にカウントダウンを続け、次アドレス“000”が入力されたとき、フラグフィールドのCFLAG“1”が出力されると同時にSSC“10”がSTDに伝達され、再びセレクタを①内部データ帰還ループから②外部入力に切換え、停止/繰り返しを継続する。この状態で任意のアドレスが外部レジスタに書き込まれると、再びカウントダウンを実行する。また、図3-9にマイクロ命令で表現した等価プログラムを併記した。ここではCFLAGの値を使って、次のアドレスに分岐するプログラムとなっている。

以上のように、ノイマン型アーキテクチャの特徴として、

- 1) PCが指定するアドレスから次の命令を読み込む
- 2) 命令長分のPCを持つ
- 3) 制御部で命令をデコードする

制御部はコンピュータの他の部分に対して命令を発行、繰り返しを行うためPCの値を替えたり、条件分岐のためにCPUの状態によってPCの値を替えたりすることができる。

が挙げられ、これらの特徴を従来のPMUモデルに取り込み、新しいシミュレーションモデルの改良を行った。

3.3.2 PMUシミュレーションモデル

PMUの基本コンセプトは、PMUを1個、すなわち最小単位の基本論理素子を使ってカウンタ/タイマ機能を実装することができることであり、ここでは、8ビットのカウンタ/タイマ機能を最小単位としたシミュレーションモデルを作成した。また、このモデルは図3-10に示すようにアドレス入力からデータ出力まで、1サイクルで動作するモデルとした。

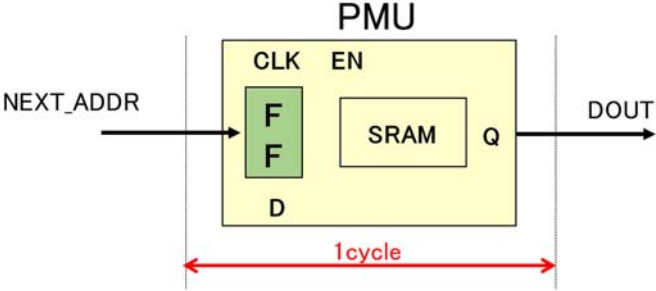


図3-10 PMUモデルの動作サイクル数

図3-11に新しいPMUシミュレーションモデルとその入出力信号を示す。図3-5のPMUアーキテクチャモデルをベースに、必要なメモリのRead/Write、モード設定、Logic動作制御信号等の制御信号を追加し、PMUのアーキテクチャと同様に、左の入力側から右の出力側に信号フローとなっており、シミュレーションモデルでも入力側に入力データ/アドレス、各入力制御信号を配置し、出力側はフラグ/データ出力とキャリー信号であるCFLAGの3種類のみとなる。

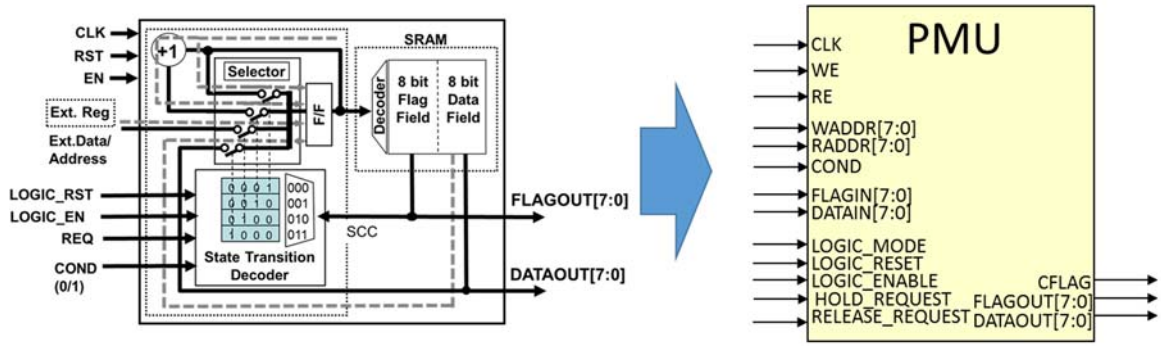


図 3-11 PMU アーキテクチャのシミュレーションモデル化と入出力信号

次に、この新しい PMU モデルを使って、初期のモデルと同様の 16 カウントのシミュレーション実験を行った。また、今回シミュレーションした 2 種類のカウンタの結線と入出力信号を図 3-12 に示す。まず始めに図 3-12(a) に示す 8 ビットのカウンタで 16 カウントを一回行った後、停止するモデルのシミュレーションを行った。そのシミュレーション波形を図 3-13 に示す。上段がシミュレーション全体の波形で、下段が 16 カウント部分の波形を拡大したものである。

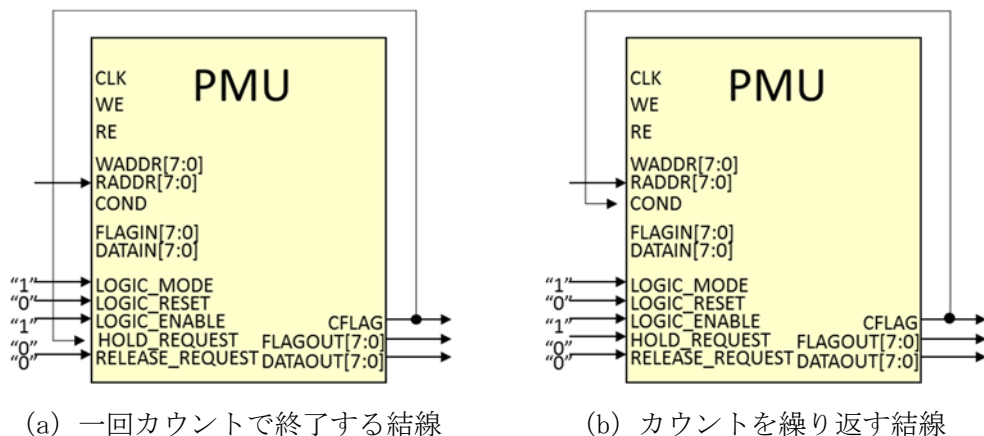


図 3-12 8 ビットカウンタモデルの入出力信号

次にこの動作について述べる。PMU はリセット後、デフォルト状態（メモリモード）で動作する。メモリモードでは、セレクタが②外部入力（図 3-8 参照）に設定されている状態が保持される。外部からの入力アドレスは WADDR[7:0] と RADDR[7:0] の 2 つがあり、それぞれ Read Enable (RE) 信号および Write Enable (WE) 信号によって SRAM の入力アドレスを選択する。これは外部からロジックモード信号が入力されない限りメモリモードは継続する。データ制御部も同様に、LOGIC_ENABLE 信号が入力されない限り制御信号は有効にならず、読み出しデータを加工せずにそのまま外部へ出力することができる。このように、メモリモード時の PMU は通常の内蔵メモリとして動作することができる。

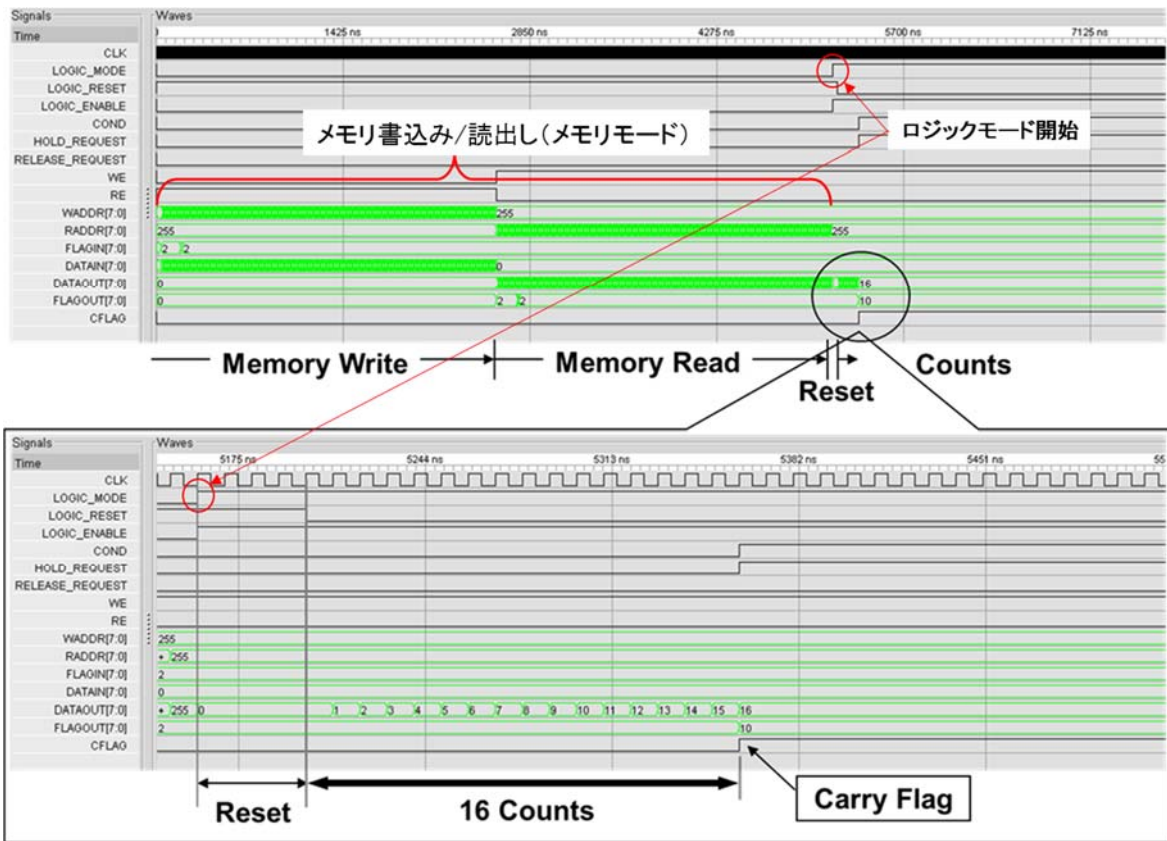


図 3-13 PMU モデルの 16 カウントシミュレーション波形 (一回)

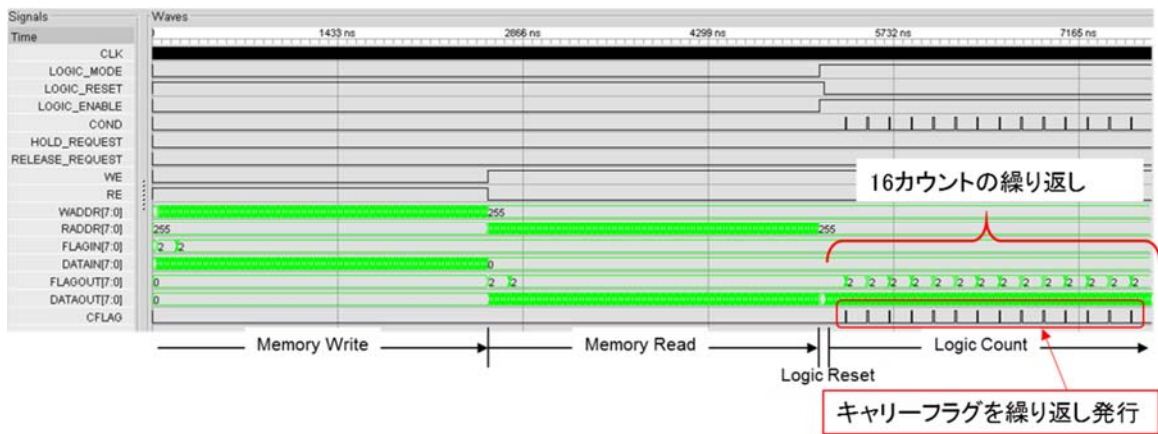


図 3-14 PMU モデルの 16 カウントシミュレーション波形 (繰り返し)

上述の如く、メモリモードで、8ビットカウンタ（256カウンタ）のコンテキストをメモリに書き込み、書き込んだコンテキストを読み出し、検証（Write-Read-Verify）を行う。次にロジックモード開始（LOGIC_MODE 信号入力）と同時に、LOGIC_ENABLE 信号が入力され、ロジックリセットが実行される。このロジックリセット時に PMU 内の FF 等の 4 サイクルのリセットが実行される。リセット完了後、自動的に 16 カウントを実行し、16 カウント完了と同時に CFLAG を発行し、カウントを停止する。

次に、図 3-12 (b) に示した 8 ビットカウンタを使って 16 カウントを繰り返し実行するカウンタのシミュレーション

ェレーション波形を図 3-14 に示す。ロジックリセット後、16 カウント完了毎に CFLAG を繰り返し発行しているのが観測され、問題なく動作することを確認した。

以上、3.2.2 で提示したコンセプトと課題および初期のモデル評価で判明した新たな検討要件に対し、対応策を施した。図 3-1 に示した初期モデルにマイクロプログラム制御方式の導入、メモリ容量の拡張等の改良を行い、PMU アーキテクチャモデルを開発し、基本演算素子モデルとした。これら実施した課題と PMU アーキテクチャでの対応策を図 3-15 に示す。

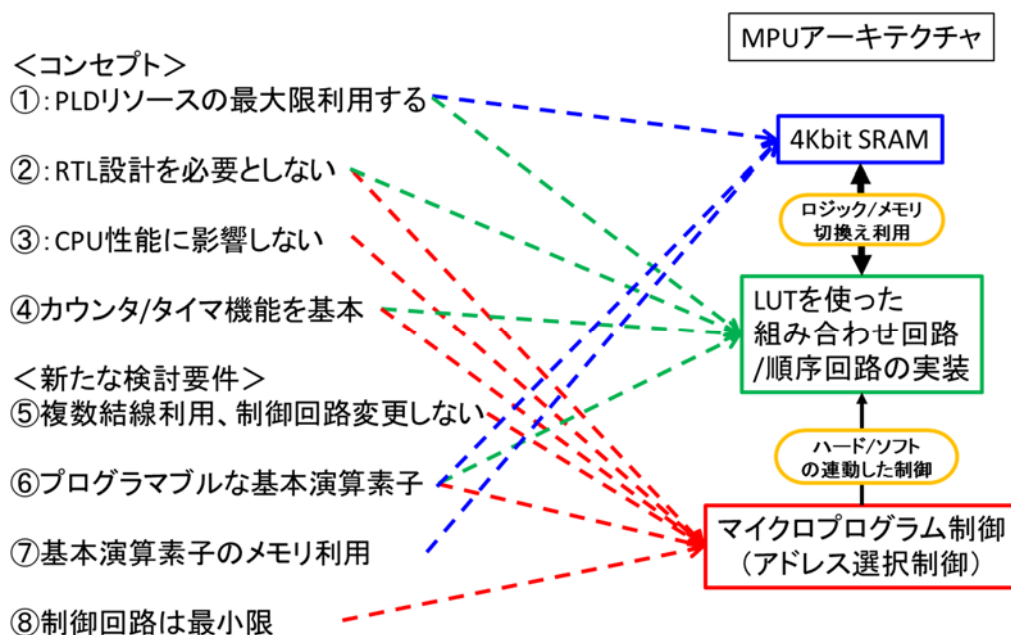


図 3-15 基本論理素子の課題と PMU アーキテクチャ

以上のように、マイクロプログラム制御方式の採用と当初の想定した 2K ビットの 2 倍である 4K ビットの粗粒度メモリを採用が最大の改善ポイントとなった。

3.4 FPSM アーキテクチャの概要

FPSM アーキテクチャは、マイコン搭載用アーキテクチャであり、マイコンのアーキテクチャに依存し、実装手段は変わる。したがって、FPSM をマイコンに搭載する場合、ノイマン型アーキテクチャとハーバードアーキテクチャの双方を考慮する必要があるが、ここではノイマン型アーキテクチャのマイコンを前提に実装、利用方法を検討した。

図 3-16 に FPSM を搭載したマイコン実装イメージを示す。ここで示すマイコンはノイマン型アーキテクチャで、マイコンの内部バスはメインバスであるメモリバスとバスステートコントローラを経由して周辺バスが接続されたマイコンである。これに FPSM 部の MCU インタフェースを介してメモリバスと周辺バスに接続されている。MCU インタフェースからはこの他に割り込み信号が出力され、マイコン本体の割り込み制御 (Interrupt Controller:INTC) に接続され、マイコン全体の割り込み制御とリンクした優先順位が割り付けられる。このような仕組みは半導体メーカーによって、イベントリンクコントローラ [3-1]

などと呼ばれている機能と同じである。マイコン内の周辺回路の割り込み信号を利用し、周辺回路同士を、CPU を介さず動作させるのに用いられる。このように、既存のマイコン製品にプログラマブルロジックデバイスを内蔵し、かつ周辺回路を実装するには必須の機能である。

このように FPSM は、基本論理素子である PMU アレイ配列部と、マイコン内蔵のための MCU インタフェースで構成されている。特に、マイコンとのインタフェースは、各種マイコンとのインタフェースのため出来るだけ一般化しやすい方式が望まれる。

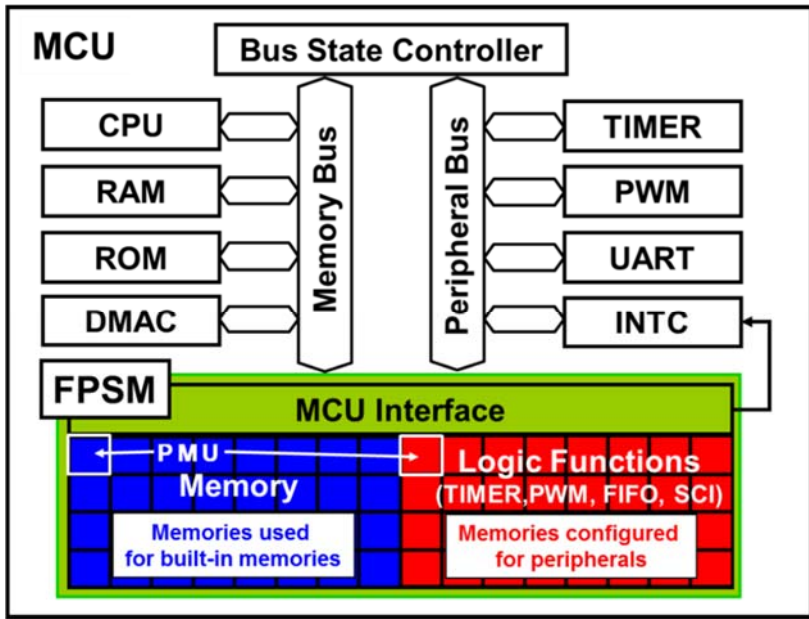


図 3-16 FPSM を搭載したマイコンの構成例（実装イメージ）

FPSM は、PMU を 2 次元に配列したアレイで構成され、PMU 同士を複数結線して、カウンタ/タイマ、PWM、FIFO、シリアル通信インタフェース等の周辺回路を実装する。また、PMU はプログラマブルロジックとして利用しない場合は、通常の内部メモリとしても利用できる。

PLD では、プログラマブルロジックとして利用しない領域は、利用されないまま放置される。例えば PLD の実装率が 40% とすると 60% のリソースが使われないままとなる。この状況に対して、FPSM では、FPSM の領域の半分に周辺回路を実装、すなわち FPSM の実装率 50% とすると、残りの 50% の FPSM のリソースは一つの連続的なメモリ領域として利用する事ができる。プログラマブルロジックとしてでは無いが、プログラマブルロジック部の領域の利用率は 100% となる。

3.4.1 PMU アレイ構成

プログラマブルロジックデバイス配置構成は、前章の図 2-8 に示すようにアイランドスタイルが一般的である。また、DRP も同様に隣接配線とデータフローを重視したアイランドスタイルが採用されている。さらに SoC 向けに LUT とインターコネクト部を階層的に配置したプログラマブルロジックデバイスも提案されている [3-2]。今回、PMU をアレイ化するにあたって以下の要件を反映することとした。

- 1) PMU（順序回路）のカスケード接続を前提としたアレイ構造

- 2) PMU 同士の接続は、レジスタ設定によるスイッチで結線
- 3) スケーラビリティ (PMU 1 個または複数個利用を自由に設定)
- 4) PMU 間のシーケンス動作の連携

図 3-17 に PMU アレイの基本構成を示す。PMU と接続経路制御回路である SB が交互にマトリックス状に配置される。この SB は入出力信号接続制御を行い、入出力信号経路を決定する。PMU は SB を介して複数接続、組み合わせが行われるとともに、SB を介して MCU インタフェースに接続される。さらに MCU インタフェースを介して、マイコンの内部バスと接続される。PMU の両側はこの SB に挟まれた状態になっている。また、両端の列の SB は入力用と出力用が準備され、単独あるいは複数の PMU を使用する場合でも、必ず入力側 SB から出力側 SB への一方向データフローとなる。

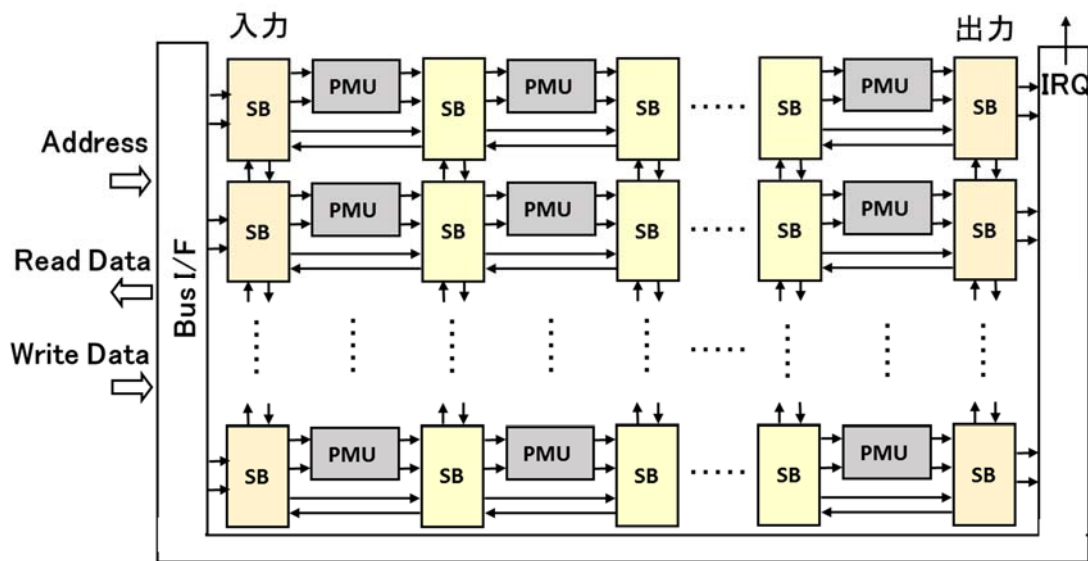


図 3-17 PMU アレイ構成

図 3-16 に示したように、FPSM は LSI への組み込み用途を前提としており、FPSM のアドレス空間は CPU 上の共通メモリアドレス空間で管理する必要がある。また、前提のマイコンではバスステートコントローラを搭載し、これを經由して周辺バスにアクセスされる。このため、内蔵メモリとして使用する場合は、メモリの出力データはマイコンのメモリバスへ、周辺回路として使用される場合は、周辺バスへデータを出力するバスの切り替え機構が必要になる。このため、メモリからのデータ出力、または論理演算の出力を分割管理出来る仕組みを導入し、従来のマイコン製品のバス構成を変更せずに追加・拡張できるようにした。さらに周辺回路を実装時には、マイコンに組み込まれた周辺回路、または外部 I/O と直接インタフェースをとることも考慮することとした。

3.4.2 MCU インタフェース

FPSM は専用の MCU インタフェースを介してマイコンのメモリバスまたは周辺バスに接続される。図 3-18 に FPSM をマイコンに実装し、メモリおよびプログラマブルロジックデバイスとして利用する場合のバ

ス接続構成を示す。FPSM はメモリとして利用する場合とプログラマブルロジックデバイスとして利用する場合の 2 つのモードを持っているため、CPU からアクセスする場合、CPU がアクセスする FPSM の領域がメモリ利用かプログラマブルロジックデバイス利用なのかを判別できなければならない。そこで、マイコンのバスステートコントローラを介して、FPSM 内でメモリとして利用しているメモリ空間、プログラマブルロジックデバイスとして利用している空間に対して、それぞれチップイネーブル信号準備してメモリまたはプログラマブルロジックデバイスとしての利用を区別する。

CPU が特定の FPSM のメモリ空間をアクセスすると、バスステートコントローラはアクセスされたメモリ空間がメモリ利用に設定されている場合は、メモリーイネーブル信号（以下、CME）を、プログラマブルロジックデバイス利用に設定されている場合はペリフェラルイネーブル信号（以下、CPE）を発行し、図 3-18 に示すルートで CPU はメモリまたはプログラマブルロジックデバイスにアクセスする。以上のようにバスステートコントローラによってアクセスできるメモリ空間をそれぞれイネーブル信号で管理する仕組みとした。

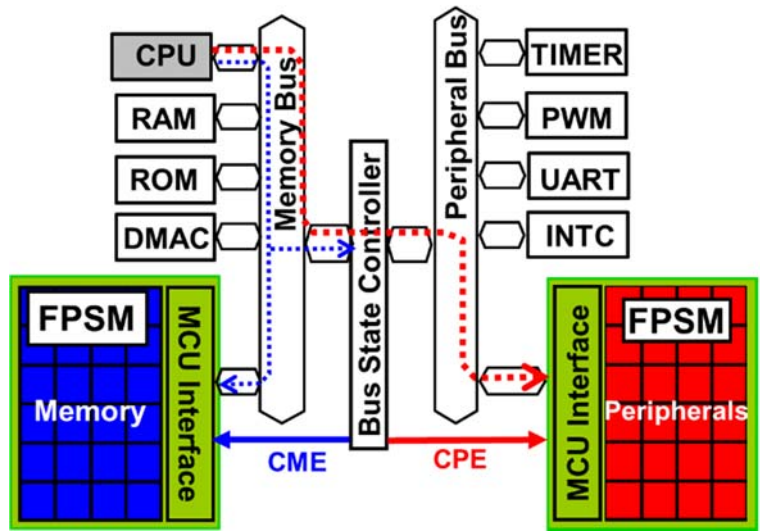


図 3-18 FPSM 実装時のバスアーキテクチャ

FPSM のデフォルト状態では、メモリモードに設定されており通常の内蔵メモリとして使用される。これはプログラマブルロジックデバイスとして利用する場合でもマイコンのブート時に、これらプログラマブルロジックのコンテキストをメモリモード状態でロードするためである。

3.4.3 FPSM のメモリ管理とアクセス方法

次に、FPSM のメモリ空間管理とアクセス方法の詳細について述べる。PMU のメモリ構成は 16 ビット × 256 ワードに決定したが、この FPSM 内の PMU は全てローカルな 256 ワードのアドレスになっている。マイコンはグローバルアドレスで FPSM をアクセスするため、FPSM 内の PMU の場所を特定するためにアドレスをグローバルアドレスから FPSM 内のローカルアドレスに変換する必要がある。そこで FPSM の MCU インタフェース内に PMU のマトリックス状に配置された X-Y 座標情報を持つ PMU アレイ座標変換部を準備し、この X-Y 座標情報を使ってマイコンのグローバルアドレスで管理するようにした。これによりグロ

グローバルアドレス⇄ローカルアドレス変換が可能になる。

すなわち、

グローバルアドレス=8ビット座標情報 (CX, CY) +8ビットのローカルアドレス

となる。座標情報は8ビット以内で自由に設定できるため、マイコンに実装するPMUアレイ構成に合わせてビットを割り振ればよい。マイコンのチップ面積、バリエーション等に対してもフレキシブルに対応可能なスケラビリティを持っている。

FPSMのメモリ空間は、マイコンの共通アドレス空間で管理される。マイコンの共通アドレス空間内に、FPSMのメモリ空間が割り当てられた状態を図3-19に示す。

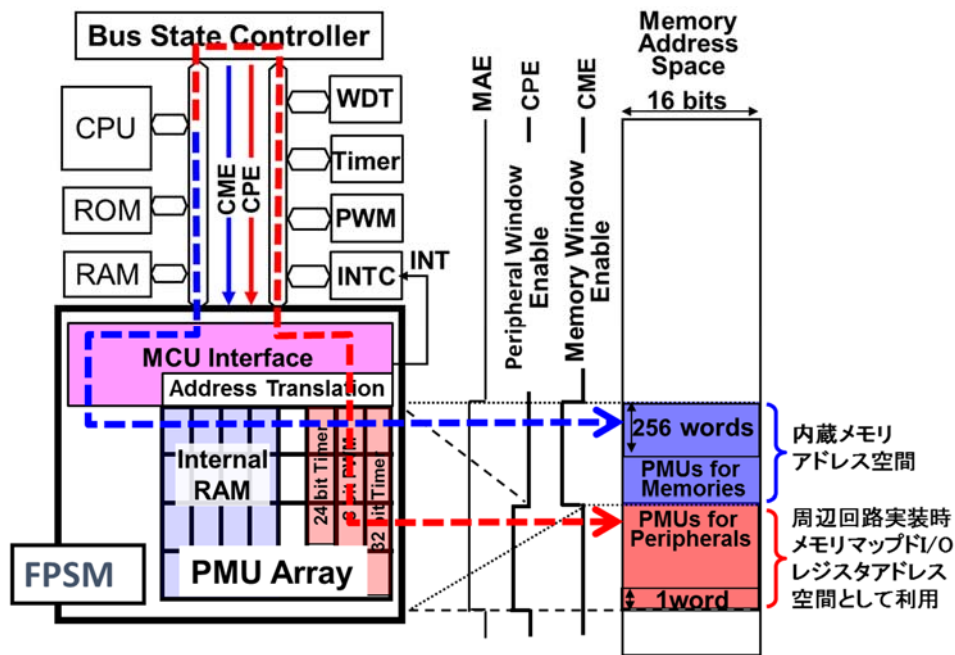


図 3-19 FPSM のメモリ空間管理

MCU インタフェースは、マップド I/O 方式で FPSM のプログラマブルロジックデバイスに実装された周辺回路のアドレスを管理する必要がある。CPU からはプログラマブルロジックデバイス上に実装された周辺回路は、マイコンのメモリ空間にマップされたレジスタアドレスとして割り振られるメモリマップド I/O 方式のため、FPSM をプログラマブルロジックデバイスと利用する場合は、実装される PMU の先頭アドレスを使って、レジスタアドレスと同様の 1 ワードのグローバルアドレスとして利用する。PMU を複数使う場合も、同様にこれら複数の PMU の先頭アドレスを使った 1 ワードがグローバルアドレスとして割り当てられる。以上によりプログラマブルロジックデバイス上に実装された周辺回路をメモリマップド I/O として利用することができる。

FPSM ではマイコンが持つ共通のメモリ空間上に FPSM の 2 倍のメモリ空間を確保する。これは FPSM を全てメモリとして利用する場合のメモリ空間と、FPSM を全てプログラマブルロジックデバイスとして利用する場合のメモリ空間を確保する。すなわち、実装する FPSM の 2 倍のメモリ空間を準備して使い分ける方式とした。これは、管理のしやすさに加え、上書き防止も兼ねている。

CPU が FPSM のメモリ空間にアクセスする場合は、マイコン上のメモリ空間に割り当てられた FPSM の全メモリ空間に対して Memory Access Enable (MAE) を発行し “High” にする (図 3-19). さらにメモリとしてアクセスする場合、バスステートコントローラはアクセスされたメモリ空間がメモリに設定されている場合は、バスステートコントローラから CME 信号が発行されメモリ利用空間のみ “High” になり、所望の PMU のメモリ空間をアクセスできる.

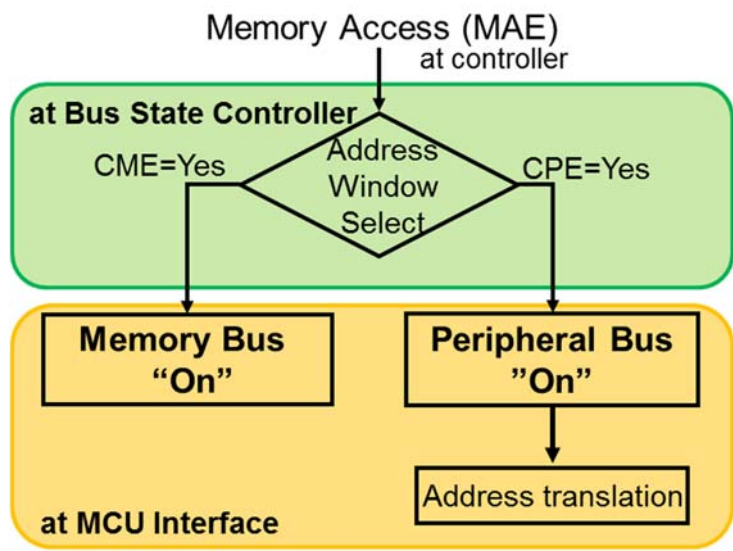


図 3-20 MCU インタフェース部分のアドレス変換 (Address Translation)

また、周辺回路が実装されたプログラマブルロジックデバイスをアクセスする場合、同様に、バスステートコントローラから CPE 信号が発行され周辺回路利用空間のみ “High” になり、所望のメモリマップド I/O のレジスタアドレスとして設定された 1 ワードのメモリアドレスを持つ周辺回路としてアクセスできる. このように、メモリと周辺回路を使い分けるために、図 3-19 に示す FPSM の MCU インタフェース部にアドレス変換を設けている. 図 3-20 にそのアドレス変換の選択方法を示す. バスステートコントローラはアクセスされたメモリ空間が FPSM のメモリ空間か周辺回路空間かを判定し、MCU インタフェースは CME か CPE の信号を使ってバス選択を行う. また、周辺回路利用空間する場合は、必ず PMU の先頭アドレス単位でアクセスできるようにアドレス変換部を通過して所望の PMU にアクセスすることができる.

3.5 スイッチボックス (SB)

SB は、PMU カスケードおよび並列接続用の経路選択回路である. 図 3-21 にこの SB の概念モデルを示す. SB は前段の PMU からの入力信号を入力セレクタで後段の PMU に出力するための経路選択を出力セレクタで行う. PMU 同士の信号のやり取りを行うローカル配線と SB 同士の信号のやり取りを行うグローバル配線の 2 種類が用意されている. ローカル配線は一方向、グローバル配線は双方向のデータフローになっている. 上部の SB と下部の SB との接続はこのグローバル配線を使って接続される. これら PMU 間の経路情報は、各 SB 内に準備されたレジスタに格納され、SB 内の経路選択スイッチを制御している.

また、図 3-17 に示した出力側の SB には CPU への割り込み信号を出力するステータス部が設けられ、プログラマブルロジックデバイスで周辺回路を実装した時に用いられる。

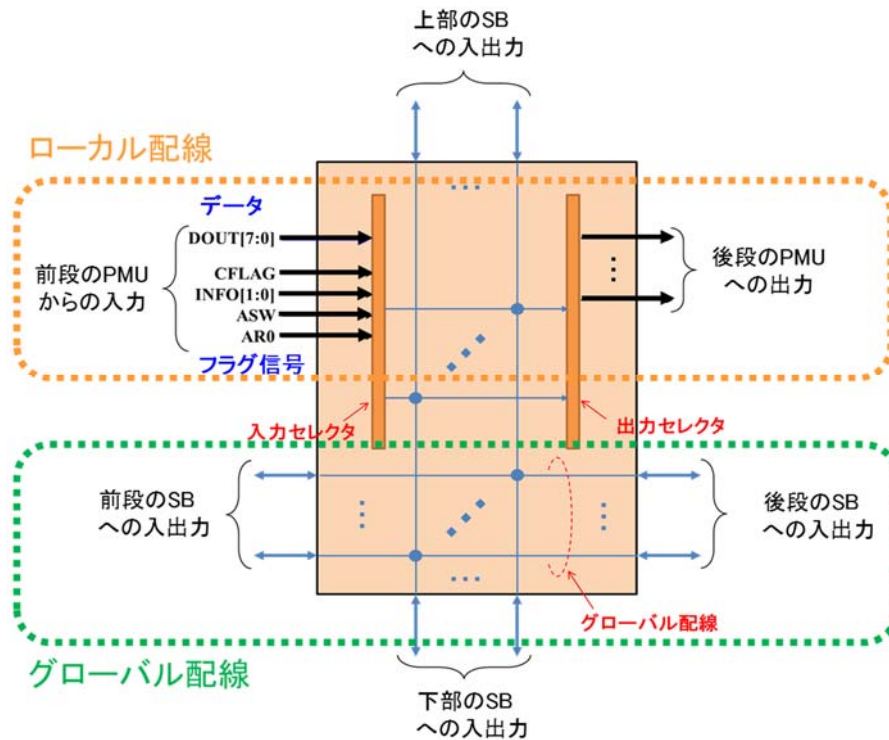


図 3-21 SB の概念モデル

図 3-22 に SB モデルの回路構成を示す。基本的に PMU が複数個接続される場合は、PMU はデータフィールドの 8 ビットと CFLAG[1:0] のうち 1 ビットおよび SEQ (SCC から信号名変更) の 3 ビットを合わせた 4 ビットの制御信号の計 12 ビットの信号を他の PMU に送信する。そこで、柔軟性を保ちながらハードウェア量を抑えるため、データフィールドのデータを上位 4 ビット、下位 4 ビットに分割およびフラグ信号の 4 ビット全てを 4 ビット幅に標準化して入出力信号を取り扱う方式とした。

SB には 4 ビット幅の 4 本のグローバル配線が接続され、SB に設けられたバススイッチを経由し、他の SB を介して所望の PMU と接続される。SB は前段の PMU からの出力である 4 ビット幅の信号 3 本 (IN1, IN2, IN3) と上部 North SB からの 4 ビット幅の 1 本の信号が入力される。これらの中から所望の信号を入力セクタで選択し、グローバル配線に接続される。このグローバル配線のうち 1 本は下部 South SB への接続用に使われる。また、PMU のカスケード接続をする場合、前段の入力 (IN1, IN2) をダイレクトに次段の PMU に転送する場合を考慮し、2 本のローカル配線が入力セクタから出力セクタに直接接続されている。出力セクタはグローバル配線からの 2 本の信号と前段の PMU から入力セクタに入力した信号の内の 2 本の出力を選択し、計 4 本の出力信号を次段の PMU に出力する。加えて、出力セクタは、FIFO 機能などの制御に必要な 4 ビット制御信号の 1 本、合わせて 5 本の出力を制御する。これらは、SB 内に準備された 8 ビットレジスタ 4 本で設定される。

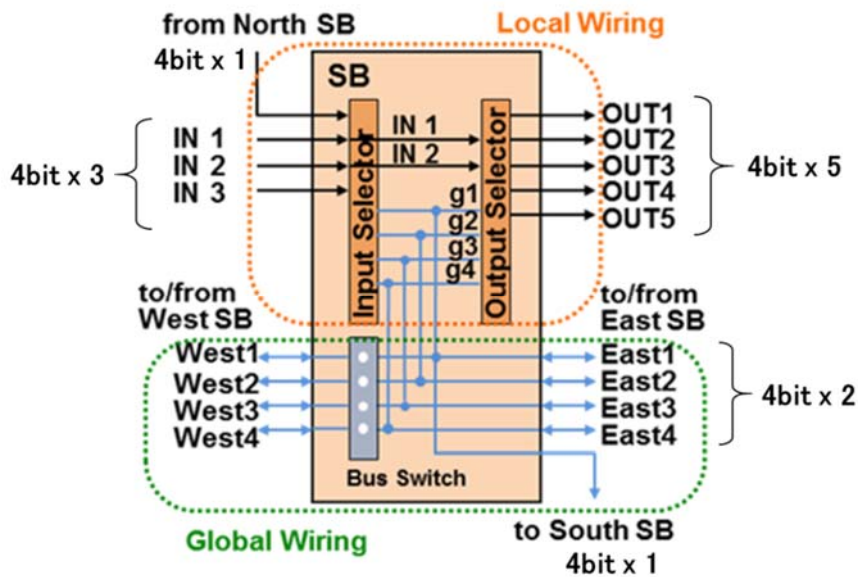


図 3-22 SB の回路構成

また、図 3-23 にこの SB を用いて、PMU アレイ上でメモリモードとロジックモード時の周辺回路を実装して利用する場合の結線、動作イメージを示す。メモリモードでは、隣接の PMU をカスケードに結線して利用する。ロジックモードでは、PMU 同士だけでなく SB 間での結線も行われ、ローカル配線とグローバル配線を駆使し、上下の行や、隣接の PMU をスキップして他の PMU と結線することで所望の機能を実装することができる。

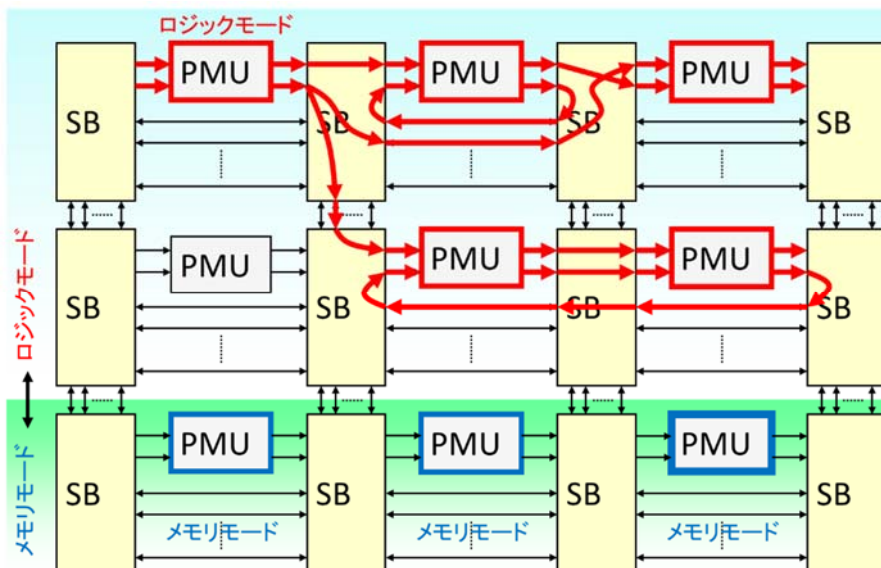


図 3-23 PMU アレイの結線と動作イメージ

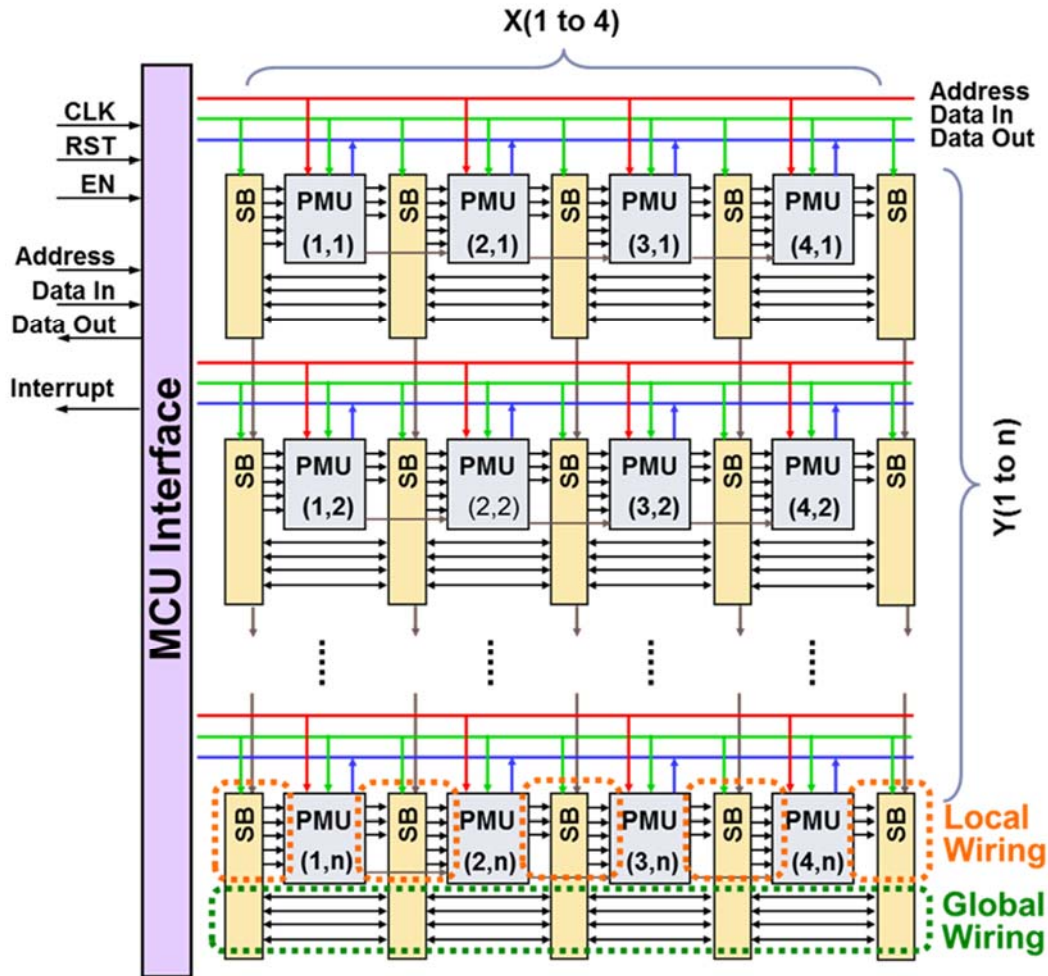


図 3-24 FPSM のブロックダイアグラム

図 3-24 に FPSM のブロックダイアグラムを示す。4 行×N 列の PMU 構成とした。一般の 8～32 ビットマイコンで利用される周辺回路およびメモリは、8 ビットから 32 ビットのワード長で利用される。PMU は 1 個で 8 ビットの基本論理素子を実現することができ、最大 32 ビットのワード長の周辺回路/メモリを実装するため、PMU を 4 個/1 行とした。これを一つのグループとし、PMU を 1 個から 4 個までのカスケード接続あるいは 4 個×N 並列に任意に接続できる構成とした。各行にはアドレスバス、入力データおよび出力データ用バスの 3 本が配置され、PMU と接続されている。SB では入力データバス以外は他の SB からの入力と PMU のローカル配線とグローバル配線で結合される。

図 3-25 に 16 ビットのカウンタを実装した場合のイメージ図を示す。PMU を 2 個使い、PMU①は下位の 8 ビットカウンタ、PMU②には上位 8 ビットのカウンタが実装される。PMU①の CFLAG が 256 カウント毎に出力される。この CF が SB を経由して、PMU②のイネーブル信号 (EN) として入力にされると、PMU②が起動され、1 カウントを実行、これを繰り返すことで 16 ビットのカウンタとして動作する（詳細は第 4 章で述べる）。

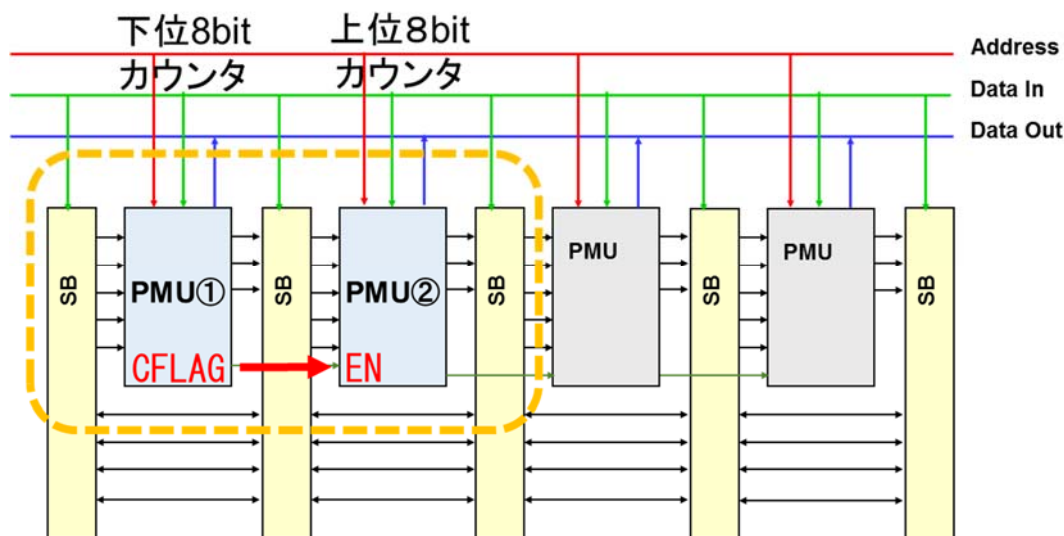


図 3-25 PMU アレイで 16 ビットカウンタを実装する例

また、図 3-17 の説明でも述べたが、このアーキテクチャの特徴から PMU の単独、あるいは複数接続する場合、一方向のデータフローとなるため、PMU の両端に入力用と出力用の SB を配置している。さらに、16 ビット精度の PWM を実装する場合、一部の信号は上下、左右の PMU と SB を経由して PMU の信号を送受信する必要があるため、SB と PMU はサンドイッチ状に配置される。

PMU アレイに MCU インタフェースが付加され、これを介してマイコンのメモリバスあるいは周辺回路バスと接続される。マイコンとの信号のやり取りは、すべてこの MCU インタフェースを経由して行われる。上述のように、MCU インタフェースではマイコンとのアドレス変換や 16 ビットの入力データバス、出力データバスとの配線、タイミング調整などが行われる。

3.6 結言

本章では、マイコン向けプログラマブルデバイスを実現するための鍵となる基本論理素子である PMU のアーキテクチャを、SystemC のモデルベース設計手法を用いて開発した。この PMU は従来の PLD の基本論理素子とは違い、粗粒度のメモリを用いマイクロ命令によって小規模なシーケンスプログラムが動作する。CPU の PC と同様に、マイクロ命令を使ったアドレス制御が可能であり、シーケンシャル動作、条件分岐/無条件分岐命令を実行し、プログラムされた動作とメモリ出力を行うことで、特定の機能を実現（模擬）できる。ここでは、カウンタ/タイマ機能を実装し、モデルシミュレーションによる動作検証を行った。また、PMU を複数結線するための SB とアレイ構成およびマイコンに内蔵するための MCU インタフェースを合わせ込んだ FPSM アーキテクチャを開発した。PMU アレイは、PMU×4 個/1 行とし、N 列並べた構成となっており、基本的に 1 行毎に機能実装を行うこととした。これにより 8 ビットから 32 ビットまでのワード長に対応する機能が自由に実装可能になった。例えば、16 ビットのカウンタを実装する場合は、PMU×2 個、32 ビットカウンタを実装する場合は PMU×4 個といった具合に、スケラブルに実装可能となった。さらにこれら PMU は 4K ビットの SRAM を想定しており、プログラマブルロジックとして利用しない場合は、PMU を 256 ワード×16 ビットのメモリとして利用できる。

また、マイコン内蔵時にはマイコンの共通メモリ空間で管理されなければならない。FPSM をメモリとして利用する場合は、グローバルアドレスで、プログラマブルロジックに実装された周辺回路はメモリマップド I/O 方式で管理する。このため、MCU インタフェース部にアドレス変換機能を追加することで対処した。これによりマイコンユーザは特に意識をすることなく、これまでの利用方法でこのプログラマブルロジックである FPSM を利用する事ができる。以上のように、マイコン周辺回路をフィールドプログラマブルに実装可能なアーキテクチャをシミュレーションモデルで開発し、実現可能である見通しを得た。

参考文献

- [3-1] <https://www.renesas.com/> “ルネサス マイクロコンピュータ RX210 グループ ユーザーズマニュアル ハードウェア編”
- [3-2] H. Nakano, T. Iwao, T. Hishida, H. Shimomura, T. Izumi, T. Fujino, Y. Okuno, K. Arimoto, “An embedded programmable logic matrix (ePLX) for flexible functions on SoC,” Proceedings, IEEE Asian Solid-State Circuits Conference, pp.219-222, Nov. 2006.

第4章 FPSM モデルシミュレーションと FPGA 実装による評価

4.1 緒言

本章では PMU を複数組み合わせ合わせたマイコン周辺回路機能の動作モデルを構築し、評価した結果を述べる。SystemC によるモデルベース開発手法を使って、内蔵メモリとしての動作確認、PMU のシミュレーションモデル上にカウンタ/タイマ系、シフトレジスタ系および演算系の回路を実装し、動作確認を行った。この過程で、モデルベース開発のメリットでもあるシミュレーションを繰り返すことで、PMU のマイクロ命令/アドレス制御方式だけでなく、入出力 I/O、設定条件等も含むアーキテクチャの改良を行った。

次に PMU と SB によるアレイ構成で実装される代表的なマイコン周辺回路機能として FIFO、シリアル通信インタフェース、PWM のシミュレーションモデルを構成し、動作確認を行った。

また、上述のシミュレーション検証を行った機能モジュールの中から 8 ビット PWM 機能を選び、FPGA 実装を行い、波形観測を行った。市販の FPGA ボードを使用し、3 個の PMU を使って 8 ビット PWM を、また、起動信号、HOLD/RELEASE のトリガ入力およびパルス幅発生用アップダウンカウンタを PMU 1 個追加実装し、PWM の動作確認、波形観測を行った結果を述べる。

4.2 PMU モデルと各周辺回路シミュレーション

これらシミュレーション環境は上述の如く、SystemC 2.1.v1、波形出力は、GTK Wave 3.0.19 (Windows XP SP2) で行った。PMU アレイに周辺回路の実装を行う場合の手順は、

- ①リセットの実行：PMU と SB の全レジスタをリセット（デフォルト状態はメモリモード）
- ②次に PMU のメモリ部に実装する周辺回路の真理値表データおよびフラグデータをデータフィールドおよびフラグフィールドに書き込む（コンテキストのロード）
- ③初期条件として、メモリ/ロジックモード選択、8/16/32 ビットのワード長設定および SB の経路選択設定情報が各レジスタに書き込む

この 3 ステップにより、実装されたコンテキストで周辺回路機能が FPSM 上に構成され、初期のスタンバイ状態になる。さらに、起動トリガとして、

- ④外部イベント (Enable 信号) の入力 (クロックは事前に供給されている前提)

によって自律的に動作が開始される。図 4-1 にこのフローチャートを示す。マイコンに内蔵して利用する場合は、これらコンテキストは、マイコンに内蔵されているフラッシュ ROM 等の不揮発性メモリに格納され、マイコンの起動時に実装される事を想定している。

以上のシミュレーション環境および実装方法で、マイコンに利用される基本論理演算回路のモデリングとその評価を行った。各基本論理演算回路で、先に準備した PMU のマイクロ命令、SB の経路選択制御の動作確認をしながらこれらの改良を行った。

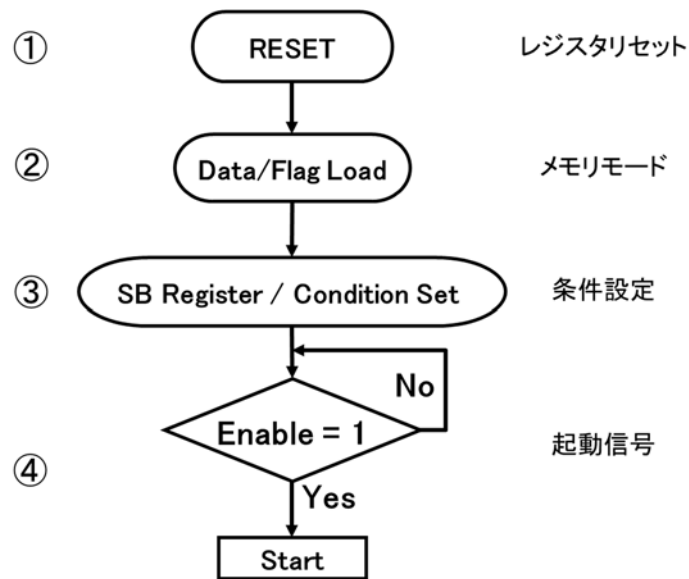


図 4-1 FPSM の実装/設定フローチャート

4.2.1 基本論理演算のモデリングと評価

PMU アレイ構成としてのメモリ機能の確認を行うとともに、基本論理演算回路として、カウンタ/タイマ、シフトレジスタおよび算術演算を選び、プログラマブルにビット長を変更して論理演算回路の動作検証を行った。ターゲットの論理演算回路を表 4-1 に示す。これら基本論理素子をライブラリ化し、実装部品の最小単位として利用することで、マイコン周辺回路やマイコンに外付けされる簡単なロジック回路をマイコンに取込むことも可能となる。具体的にはメモリ機能を含む各基本論理演算を、ワード長 8/16/32 ビットをプログラマブルに可変、実装して評価を行った。

1) メモリ機能

- ・ランダム/シーケンシャルアクセスを 8/16/32 ビットで評価（通常の内蔵メモリ機能を想定）

2) カウンタ/タイマ機能

- ・アップ/ダウンカウンタを以下の仕様で評価
 - (1) 指定した回数を繰り返すカウンタ
 - (2) 任意の値で停止するカウンタ
 - (3) 条件信号で、カウント停止/再開ができるカウンタ
- 以上のような、フレキシビリティのあるカウンタを想定

3) シフトレジスタ機能

- ・論理シフト
 - (1) 1/2 ビット左シフト
 - (2) 1/2 ビット左ローテート
 - (3) 1/2 ビット右シフト
- ・算術シフト
 - (1) 1/2 ビット左シフト
 - (2) 1/2 ビット左ローテート

4) 演算器

- (1) 8/16 ビット加算器
- (2) 8 ビット減算器

表 4-1 基本論理素子を使って実装する機能

Basic Function		Option	Word Length		
			8bit	16bit	32bit
Memory	Random Access	Sequential Access	✓	✓	✓
Counter /Timer	Up Count	Repeat	✓	✓	
		Arbitrary		✓	
		Hold & Release	✓	✓	
	Down Count	Repeat	✓	✓	
		Arbitrary		✓	
		Hold & Release	✓	✓	
Shift Register	Logic shift	1bit Left	✓	✓	
		2bit Left	✓		
		1bit Rotats(Left)	✓	✓	
		2bit Rotats(Left)	✓		
		1bit Right	✓	✓	
		2bit Right	✓		
	Arithmetic shift	1bit Right	✓	✓	✓
		2bit Right	✓		
		1bit Rotats(Right)	✓	✓	
		2bit Rotats(Right)	✓		
Calculation Unit	Adder	-	✓	✓	
	Subtracter	-	✓		

これらは、8/16 ビットマイコンを意識した論理演算回路であり、様々な応用に利用される。また、メモリを用いた他の演算器も提案されており [4-1][4-2]，算術演算回路として加算器，減算器も実装評価も実施した。冗長性が高く実用性は高くないが，メモリを利用して実装可能であることを確認するため，実験を行った。

次に，表 4-2 に PMU のマイクロ命令を示す。上述の論理演算，算術演算シミュレーション評価を行うため，今回使用するシミュレーションモデルのアドレス制御用マイクロ命令を定義した。

表 4-2 アドレス制御に用いる PMU のマイクロ命令の定義

FLAG[7:0]						Data [7:0]								備 考	選択アドレス	適用事例			
SEQ			CF			type													
7	6	5	4	3	2	1	0	7	6	5	4	3	2				1	0	
ASV	info	Rsv	W	T	0	0	0	Next Address								Jump	無条件分岐	D[7:0]	
ASV	info	0	W	T				Control Info.								Auto Increment	連続シーケンス	Address Reg. +1	
ASV	info	1	W	T				Next Address								Conditional Reset	条件分岐	External Address / D[7:0]	カウンタ
ASV	info	0	T _H	T _L				CALC _H				CALC _L				Calc. Function	演算設定	External Address	演算
FH	FL	1	T _H	T _L				COUNT _H				COUNT _L				4bit counter	条件付インクリメント	Address Reg. +1,+16	FiFo(下位4ビット)
FH	FL	FC	T _H	T _L				COUNT _H				COUNT _L					条件付デクリメント		FiFo(上位ビット)
Ctrl	Info.	0	T	1	0	0	0	Control Info.								Ext. Jump	外部無条件分岐	External Address	スルー
Ctrl	Info.	0	T	1	0	1		Reserved											
Ctrl	Info.	1	T	0	0	0	0	Branch Address								Branch False	条件分岐	Increment / D[7:0]	外部イベント待ち
Ctrl	Info.		T					Branch Address								Branch True		D[7:0] / Increment	外部イベント待ち
Ctrl	Info.	S ₀	T	1	0	0	0	Shift Address								Shift Left	シフト	External Address	左シフト
Ctrl	Info.	S ₇	T					Shift Address								Shift Right		External Address	右シフト

ここで、アドレス制御用に定義されたマイクロ命令は、

- 1) 連続シーケンス
- 2) 条件分岐
- 3) 無条件分岐
- 4) 外部無条件分岐

の4種類で、これらは利用目的、選択アドレスおよび3ビットの type フラグ[2:0]で紐づけされ決定される。また、データフィールドには「Next Address」, 「Branch Address」に加え、外部起因のアドレス/加工アドレスなどの「Control Info.」の3種類のアドレス情報利用する。

また、データ演算用に定義されたマイクロ命令は、

- 5) シフト
- 6) 演算
- 7) 条件付インクリメント
- 8) 条件付デクリメント

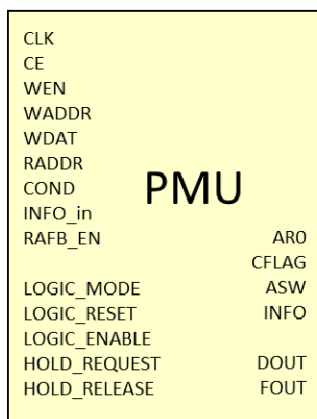
の4種類で、これらは利用目的に合わせたデータフィールド、すなわち専用 LUT と紐づけされ利用される。この Type フラグ[2:0]は、利用目的に合わせ、12種類に分類された専用 LUT を指定する。

以上、8種類の PMU のマイクロ命令を使い、各周辺回路モデルの動作検証を行った。また、前章 3.3.1 節で説明したフラグフィールドの CFLAG および SCC 信号は、拡張のためそれぞれ1ビットが追加され、

$$CFLAG: 1bit[0] \Rightarrow CFLAG: 2bit[1:0]$$

$$SCC: 2bit[1:0] \Rightarrow SCC: 3bit[2:0] \text{ (SCC から SEQ に信号名変更)}$$

に変更されている。これに伴い、シミュレーションモデルの入出力も増加したため、PMU のシミュレーションモデルと入出力を再定義した (図 4-2)。



No.	名称	数	方向	備考
1	CLK	1	IN	メモリクロック
2	CE	1	IN	セル・イネーブル
3	WEN	1	IN	ライト・イネーブル
4	WADDR	8	IN	ライト・アドレス
5	WDAT	8	IN	ライト・データ
6	RADDR	8	IN	リード・データ
7	COND	1	IN	条件信号
8	LOGIC_MODE	1	IN	論理モード
9	LOGIC_RESET	1	IN	論理リセット
10	LOGIC_ENABLE	1	IN	論理イネーブル
11	HOLD_REQUEST	1	IN	論理停止要求イベント
12	HOLD_RELEASE	1	IN	論理動作停止解除イベント
13	ARO	1	OUT	リードアドレスのFF経由最下位ビット
14	CFLAG	1	OUT	キャリー信号
15	DOUT	8	OUT	データフィールド出力
16	FOUT	8	OUT	フラグフィールド出力
17	RAFB_EN	1	IN	リードアドレス自己ループイネーブル他
18	ASW	1	OUT	アドレスループコントロール出力
19	INFO	2	OUT	ステータスセット
20	INFO_in	2	IN	条件信号2

図 4-2 PMU のシミュレーションモデルの入出力の定義

【PMU の入出力信号】

1) 入力信号

- CLK : クロック信号
- WADDR[7:0], RADDR[7:0] : 外部からのアドレス入力
書き込みアドレスを WADDR に、読み出しアドレスを RADDR に入力する
マイコンから SRAM のアドレスを指定する際に利用
- CE, WEN, WDAT : SRAM の読み出し、書き込みの制御信号
アイドル時、読み出し時、および書き込み時の設定
- COND, INFO_in[1:0], RAFB_EN : 外部条件信号の入力
これらの信号によって、アドレス制御部とデータ制御部の動作を制御
- LOGIC_ENABLE, HOLD_REQUEST, RELEASE_REQUEST : アドレスの更新/停止を制御する信号
LOGIC_ENABLE に “0”, または HOLD_REQUEST に “1” を入力することで PMU の動作は停止
- LOGIC_MODE, LOGIC_RESET : PMU のモード選択信号
以下のモード設定
メモリモード : LOGIC_MODE = “0”
ロジックモード : LOGIC_MODE = “1”

2) 出力信号

- DOUT[7:0], FOUT[7:0] : PMU からのデータ出力
FOUT はフラグフィールドからの、DOUT はデータフィールドからの出力
- CFLAG, INFO[1:0], ASW, ARO : 制御信号出力
他の PMU への制御信号に利用
ARO は現アドレス信号 RAR[7:0] の最下位ビット出力 (=RAR[0])

以上の PMU モデルと SB を使い、表 4-1 のシミュレーション実験を行った結果について以下に報告する。

4.2.1.1 カウンタモデル

表 4-3 にカウンタモデルの動作シミュレーションで利用するマイクロ命令を示す。カウンタでは条件分岐のためのマイクロ命令を用い、さらに複数の PMU 利用時のアドレス制御にも用いられる。

表 4-3 カウンタモデルで使用するマイクロ命令

FLAG[7:0]						Data [7:0]						備 考	選択アドレス	適用事例			
SEQ		CF		type													
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		
ASV	info	1	W	T	0	0		Next Address						Conditional Reset	条件分岐	External Address / D[7:0]	カウンタ

図 4-3 に 1 個の PMU に各種のカウンタを実装して基本動作の確認を行った。各カウンタの動作および PMU の結線モデルが対で示してある。アップカウンタの一回が (a)、繰り返しが (b)、ダウンカウンタの一回が (c)、繰り返しが (d) である。(a) は設定値までのアップカウント終了後、その状態がホールドされる。同様に (b) は設定値までのカウントダウン終了後、その状態がホールドされる。(b) (d) は、設定値までのアップ/ダウンカウントを繰り返す。どちらのカウンタも一回カウントと繰り返しカウントのモデルの結線は全く同じとなる。これは、カウンタ値、すなわち真理値表をアップカウンタ用、ダウンカウンタ用を準備し、実装することで実現している。

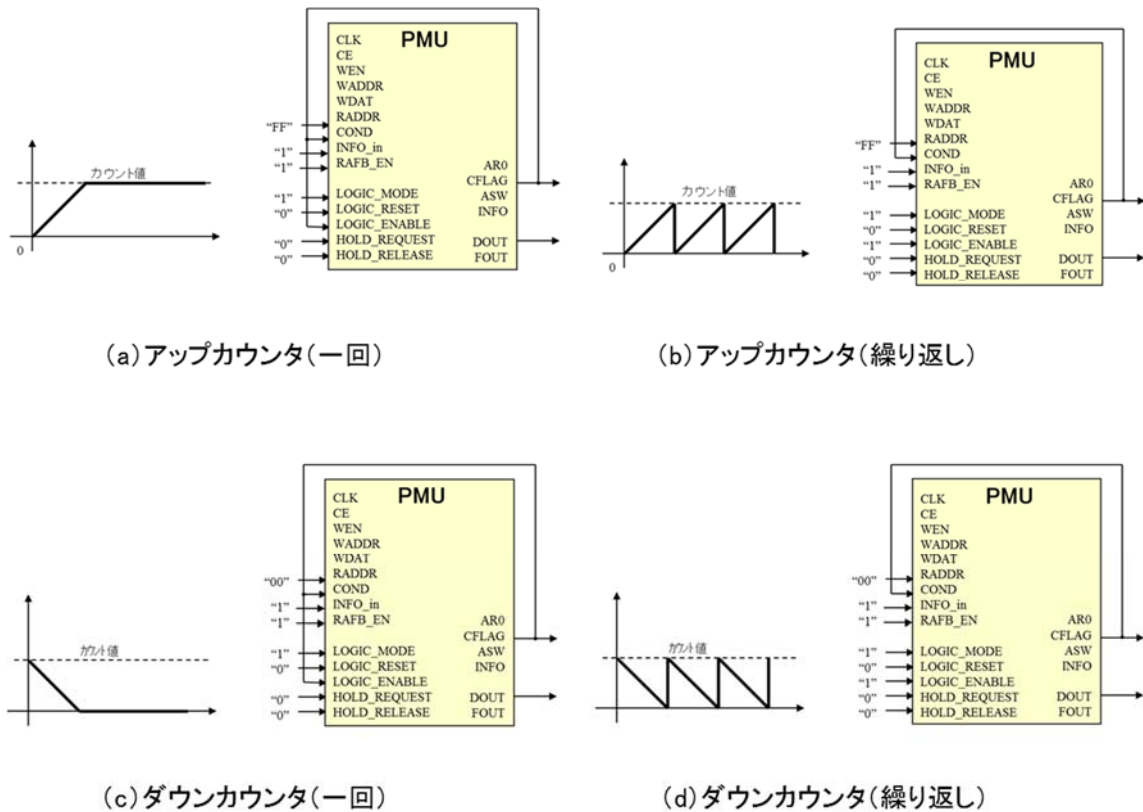


図 4-3 PMU シミュレーションモデルを使った各種カウンタの実装/配線とその動作

このように、単純な実装方法で、各種カウンタを準備でき、これらをライブラリ化（フラグフィールドとデータフィールドと結線情報）することで、実装を単純化している。

【PMU でアップカウント値の設定方法】

図 4-3 の (b) のアップカウンタ（繰り返し）を例にカウント値の設定方法を説明する。設定するカウント値=サイクル数を N とした場合、の CFLAG=1 に設定するアドレス値 A は、

- 1) 設定するカウント値は、“0” 値からのアップカウントでは、
 カウントデータ = N-1 とする
- 2) アップカウント時のアドレスとデータの関係は、
 アドレス : a, データ : d, とすると
 $a + 1 = d$
 となる
- 3) 設定するアドレス値 A は、
 $A + 1 = N - 1$
 が成立し、したがって
 $A = N - 2$
 が求められる。

以上により、N=17 の場合は、A=15 となりアドレスの 15 番目に CFLAG=1 を設定する。上述の設定方法に従い、真理値表を作成する（図 4-4）。

Function : Increment **カウント設定値のCF[0]を"1"に設定**

Input			Output				
Address			Data			CF	
DEC	HEX	BIN	DEC	HEX	BIN	1	0
0	00	00000000	1	01	00000001	0	0
1	01	00000001	2	02	00000010	0	0
2	02	00000010	3	03	00000011	0	0
3	03	00000011	4	04	00000100	0	0
4	04	00000100	5	05	00000101	0	0
5	05	00000101	6	06	00000110	0	0
6	06	00000110	7	07	00000111	0	0
7	07	00000111	8	08	00001000	0	0
8	08	00001000	9	09	00001001	0	0
9	09	00001001	10	0A	00001010	0	0
10	0A	00001010	11	0B	00001011	0	0
11	0B	00001011	12	0C	00001100	0	0
12	0C	00001100	13	0D	00001101	0	0
13	0D	00001101	14	0E	00001110	0	0
14	0E	00001110	15	0F	00001111	0	0
15	0F	00001111	16	10	00010000	0	1
16	10	00010000	17	11	00010001	0	0
252	FC	11111100	253	FD	11111101	0	0
253	FD	11111101	254	FE	11111110	0	0
254	FE	11111110	255	FF	11111111	0	0
255	FF	11111111	0	00	00000000	0	0

アドレス値 A=15 → (15, 0F) 行
 CFLAG設定 → (15, 0F) 行の CF[0] (1)
 N=17 カウント値 → (16, 10) 行

図 4-4 8 ビットアップカウンタ（繰り返し）の真理値表と CFLAG の設定

ここでは、データフィールドのカウンタ設定値 N (1 からカウンタ開始) は、

$$N = 17 \text{ とすると}$$

$$A = 17 - 2 = 15$$

となる

入力アドレスは上記の式から、アドレス値 (0 から開始) は、

$$A = 15$$

に CFLAG を設定する.

この時、CFLAG は 2 ビット [1:0] あり、このうち [0] に “1” を設定する. この時の動作タイミングを図 4-5 に示す.

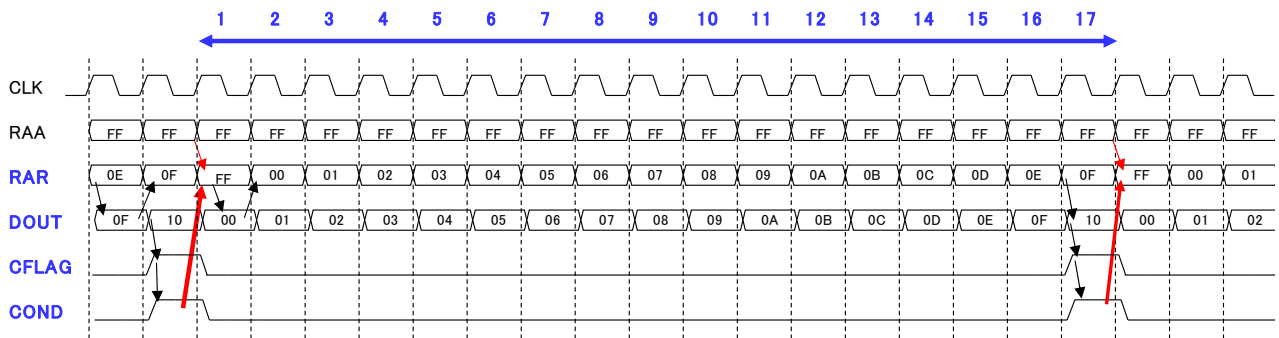


図 4-5 8 ビットアップカウンタの動作タイミング

ここで、RAR は入力アドレス、DOUT は出力カウンタ値である. N=17 でカウンタが終了すると CFLAG[0]=“1” が出力されるとほぼ同時に PMU の入力 COND に信号として入力され、FF が初期化されると再びカウンタが繰り返される. 次に、図 4-6 に 16 ビットカウンタ構成時の結線モデルを示す. 8 ビットカウンタを実装した 2 個の PMU をカスケード接続することで実現している. PMU①は下位の 8 ビットカウンタ、PMU②が上位の 8 ビットカウンタに設定されている.

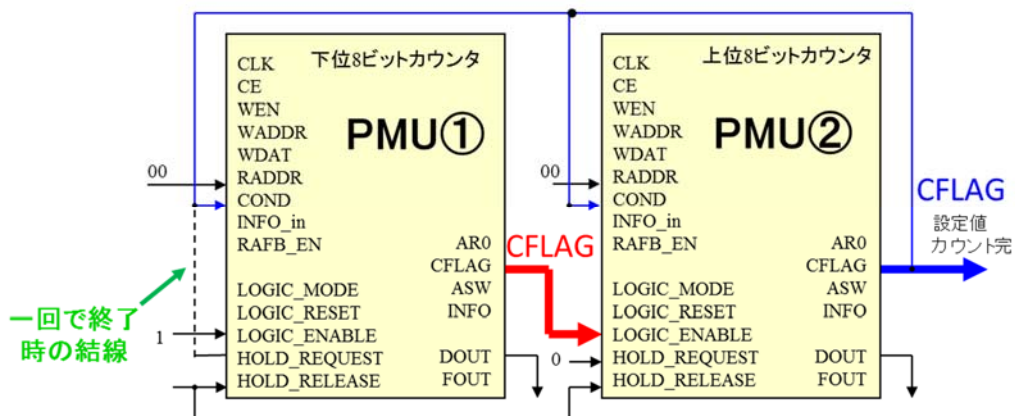


図 4-6 16 ビットカウンタ (繰り返し) 構成と結線例

この 16 ビットカウンタの動作を図 4-7 に示す。PMU①は下位の 8 ビットカウンタに設定したカウント値（最大 256 カウント）が完了すると、CFLAG が発行され、PMU①の CFLAG が LOGIC_ENABLE 信号として PMU②の LOGIC_ENABLE に入力される。この時 PMU②は、PMU①のカウント数の繰り返し回数（最大 256 カウント）が設定されており、PMU①の CFLAG 数をカウントする。PMU は自律的にカウントアップするように設定されているため、PMU②はこの外部イベントである PMU①の CFLAG が、LOGIC_ENABLE 信号として取り込み、自律的に動作が開始される。

PMU①/②に供給されるクロックと同期して、PMU②の LOGIC_ENABLE に CFLAG が 1 回/クロック入力され、PMU②は 1 クロック分だけアクティブになり、桁上げの 1 カウントを実行する。これを繰り返し、PMU②に設定された繰り返し回数、すなわち PMU①から CFLAG が出力された回数をカウントし、設定値に達したところで PMU②の CFLAG が発行され、双方の PMU の HOLED_REQUEST に入力され、カウントを停止する。また、外部から双方の PMU に HOLED_RELEASE が入力されると、動作を再開する。すなわち、最上位バイトの CFLAG 出力がコンペマッチの信号として利用可能である。

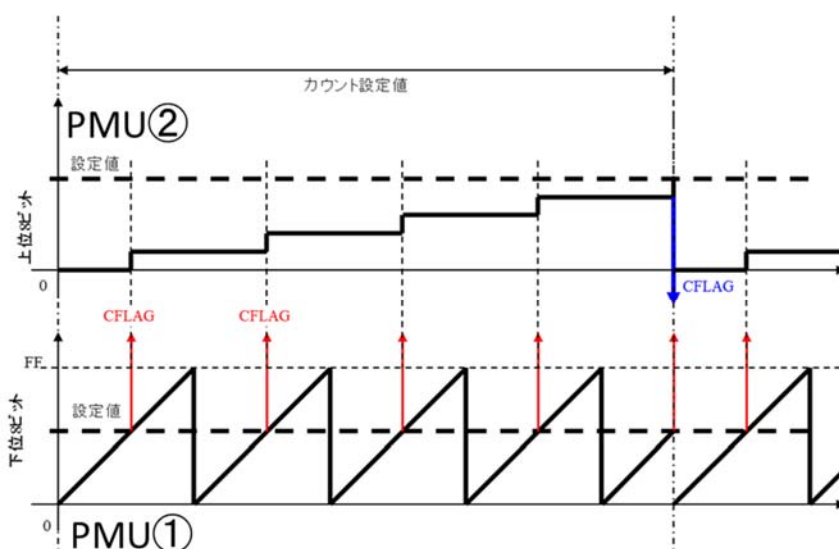


図 4-7 PMU をカスケード接続して 16 ビットカウンタを構成した場合の動作

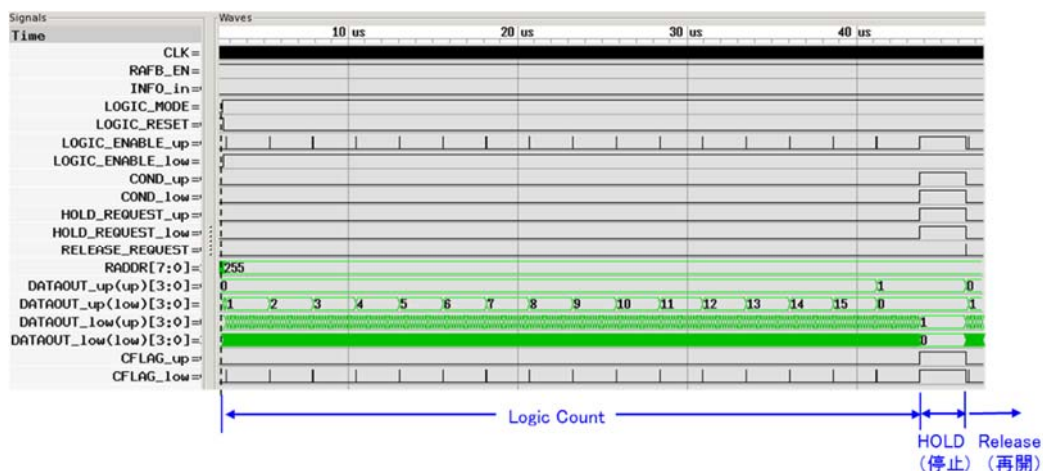


図 4-8 PMU のカウント制御（16 カウント後停止，再開）

図 4-8 にこのカウント完了後のホールド状態およびカウント再開の観測波形を示す。ここでは、16 ビットカウンタで、16 カウントを一回で停止後、カウントを再開する波形となっている。このように PMU の動作は、外部または自らイベント信号を利用して、CPU を介さず、自走するカウンタが実装できる。16 ビットカウンタを一回で終了する場合は、結線で PMU①の HOLD_REQUEST に信号に PMU②の CFLAG を入力することで実現している。また、図 4-6 に示す PMU を 2 個組み合わせさせた各種 16 ビットカウンタも同様に、問題なく動作する事を確認した。以上により、8/16 ビットの可変ワード長、アップ/ダウンおよび繰り返し、任意および停止/再開が可能なカウンタ/タイマモデルの評価を行い、問題なく動作する事を確認した。

最終的に、表 4-4 に示す 6 種類のカウンタモデルを作成し、評価を行った。8/16 ビット長、繰り返し、任意停止、一時停止/再開の機能を持つカウンタシミュレーションモデルを作成し評価を行った。

表 4-4 PMU のカウンタモデルの種類

カウンタの種類	PMU の数	
	8 ビット	16 ビット
繰り返し	1	2
任意	1	2
一時停止/再開	1	2

【イベントキャプチャ/スルー機能応用】

このほか、マイコンではカウンタ/タイマの機能を使って、様々な応用に利用するために、いくつかレジスタを追加してカウンタ/タイマ機能を強化している。その一つがコンペアマッチレジスタである。これは、割り込みを発生、出力値変更、カウンタ/タイマのクリアなどに利用される。

上述の PMU のカウンタでは、このような機能をカウント設定値に CFLAG=“1”を設定することで、コンペアマッチを行い、一時停止/再開/繰り返しなどを行うことができる。

さらに、マイコンではキャプチャレジスタを持っている。これは、外部から入力された信号、すなわち外部イベント入力時、その時点のカウンタ/タイマのカウント値を取り込むために利用する。ソフトウェア (CPU) を使って同様の動作をさせた場合、他の割り込みなどが発生すると、割り込み受付時間の変動や他のプログラム処理中のカウンタ/タイマ値の読み出しには時間がかかり、本来の外部イベント入力タイミングより遅れることになる。このためカウンタ/タイマのキャプチャ機能は重要となる。

表 4-5 にイベントキャプチャ/スルー機能モデルの動作シミュレーションで利用するマイクロ命令を示す。ここでは無条件分岐命令が用いられる。

表 4-5 イベントキャプチャ/スルー機能モデルで使用するマイクロ命令

FLAG[7:0]						Data [7:0]								備 考	選択アドレス	適用事例		
SEQ	CF		type			7	6	5	4	3	2	1	0					
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			
Ctrl Info.	0	T	1	0	0	Control Info.								Ex.t. Jump	外部無条件分岐	External Address	スルー	

図 4-9 にイベントキャプチャ/スルー機能モデルと、図 4-10 にイベントキャプチャ/スルー機能の真理値表と動作を示す。PMU①には 8 ビットの自走カウンタが実装され、常時カウンタが動作し、カウント値が Dout から出力される。この時、キャプチャトリガが外部から PMU②の LOGIC_ENABLE に入力されると、PMU②が起動し、この起動時に PMU①から出力されている Dout の出力データをアドレスとして取り込む。また、PMU②にはスルー機能用の真理値表が実装されている。この真理値表は図 4-10 に示すように、入力アドレスと同じ値がデータフィールドに実装されている。

すなわち、キャプチャトリガ入力時 = (t) とすると

$$\begin{aligned} \text{PMU①のカウンタ出力値 (t)} &= \text{PMU②の入力アドレス値 (t)} \\ &= \text{PMU②の出力値 (t+1)} = \text{キャプチャ結果} \end{aligned}$$

となり、キャプチャ結果は、1 サイクル遅延して出力される。このように、PMU を使うことで、単純なカウンタ/タイマを実現するだけでなく、機能を付加した多機能カウンタ/タイマが実現できる。

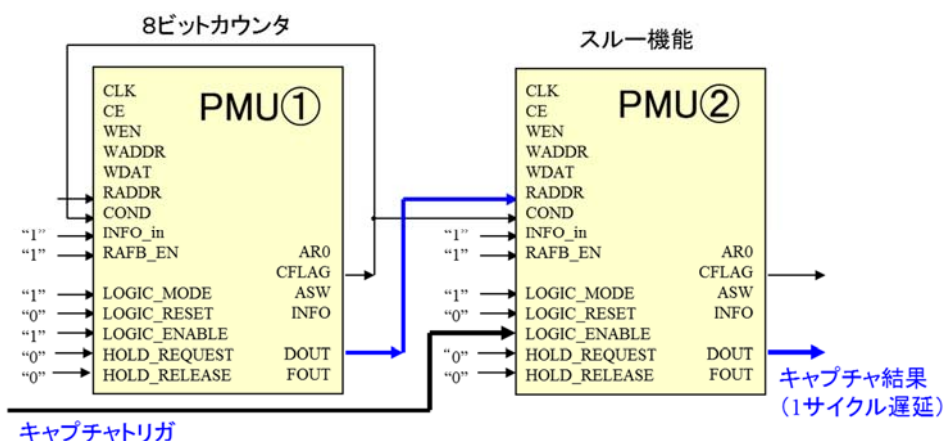


図 4-9 イベントキャプチャ/スルー機能モデル

PMU ①			PMU②	
Address	Data[7:0]		Address	Data[7:0]
.....
1000 0110	1000 0111	Dout	1000 0110	1000 0110
1000 0111	1000 1000		1000 0111	1000 0111
1000 1000	1000 1001		1000 1000	1000 1000
1000 1001	1000 1010		1000 1001	1000 1001
.....

図 4-10 イベントキャプチャ/スルー機能の真理値表と動作

以上のように、マイコンにおけるカウンタ/タイマの応用範囲は広く、また、このスルー機能により、外部イベントのカウンタ、パルス幅測定、時間間隔測定などに利用可能である。さらに、インターバルタイマや方形波等のパルス出力、PWM 出力等々、様々な用途に活用される。

4.2.1.2 シフトレジスタモデル

表 4-6 にシフトレジスタモデルの動作シミュレーションで利用するマイクロ命令を示す。ここではシフトレジスタの基本命令として左シフト命令、右シフト命令が用いられる。

表 4-6 シフトレジスタモデルで使用するマイクロ命令

FLAG[7:0]						Data [7:0]						備考	選択アドレス	適用事例				
SEQ			CF			type												
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			
Ctrl Info.	S ₀	T			1	1	0	Shift Address						Shift Left	シフト	External Address	左シフト	
Ctrl Info.	S ₇	T					1	Shift Address						Shift Right		External Address	右シフト	

PMU で 1 ビットシフトレジスタの基本動作を表現すると、

- ・入力されたアドレス（またはデータ）に対し、左/右にシフトした情報を PMU のデータフィールドから出力する

となる。そこで、データフィールドの真理値表と CF[1:0] を PMU の入力 COND の入力信号として利用する事でこれを実現する。

PMU の入力 COND に入力する信号によってデータフィールドの S₀, S₇ として定義した LSB, または MSB に埋め込むデータを変更することで実現する (表 4-7)。これにより、論理シフト、算術シフト、およびローテートが実現することができる。また、CFLAG をカスケード接続することで、シフト量 (ビット幅) を拡張することができる。

表 4-7 PMU の入力 COND に入力するキャリー信号とデータフィールドの制御

	CF[1]	CF[0]
左シフト	S ₀ : 出力データの LSB 置換イネーブル	T: 入力アドレスの MSB データ
右シフト	S ₇ : 出力データの MSB 置換イネーブル	T: 入力アドレスの LSB データ

図 4-11 に一般的なシフトレジスタ構成を示す。シフトレジスタは FF を構成したいビット数分並べて n ビットシフトレジスタが構成される。また、並列/直列入出力のタイプがあるが、ここではシリアル入力/シリアル出力のタイプでの実装を行った。

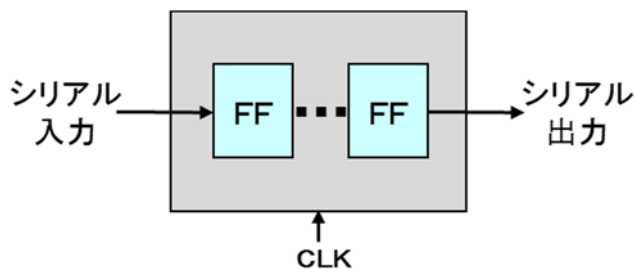


図 4-11 一般的なシフトレジスタの構成

PMUで1ビットシフトレジスタの基本動作を表現する場合、入力する1ビットシフトしたい値（アドレス）が入力し、このアドレス値が1ビット左シフトした値を出力すればよい。

これをPMUに実装する場合は、メモリのアドレスとデータ、アドレス制御を使って実現する。この場合は、左右1ビットシフトレジスタの真理値表、Cond信号およびCFLAG信号を利用して実現する。図4-12にPMUを使った左右1ビットシフトレジスタの動作モデルを示す。

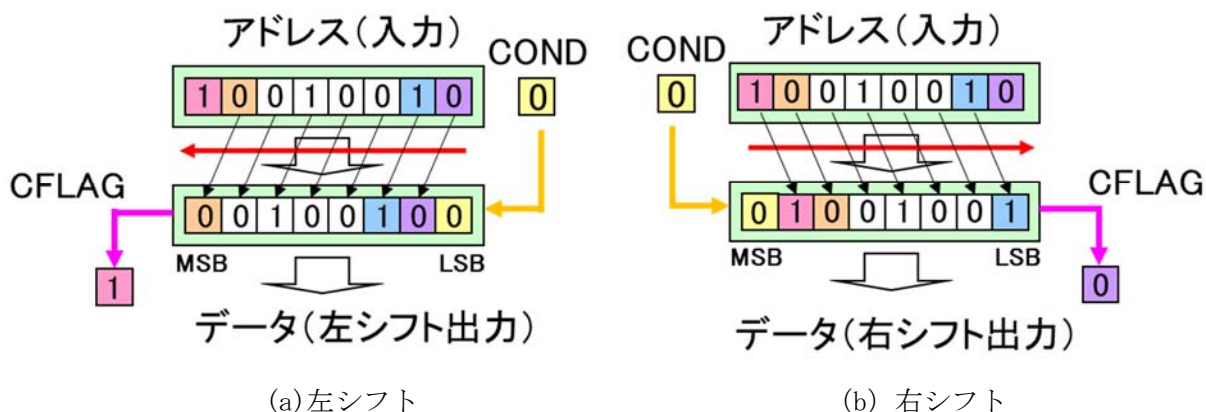


図 4-12 PMU のシフトレジスタの動作モデル

図4-13にPMUの1ビットシフトレジスタのシミュレーションモデルを示す。左、右シフトレジスタと同じモデルであり、実装する真理値表で機能が決まる。

図4-14にこのモデルに実装する左1ビットシフトレジスタモデル (a)、右1ビットシフトレジスタモデル (b) の真理値表を示す。これらの真理値表は、各シフト専用で作られたもので、この真理値表にある情報を利用して、出力値のMSB/LSBのビットを加工して出力値を生成する。ここでは4ビットの真理値表を使って説明する。

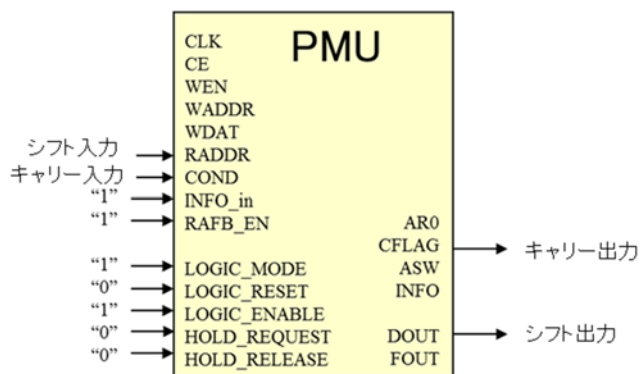


図 4-13 PMU の1ビットシフトレジスタのシミュレーションモデルと入出力

図4-12 (a) の左1ビットシフトの真理値表において、PMUでは上述の如く、入力アドレス値に対して、左1ビットシフトした値を出力すればよい。この真理値表の入力アドレスで、左1ビットシフトすると

桁上げ/キャリーが発生するグループ (CFLAG=“1”), ならびに発生しないグループ (CFLAG=“0”) がある. この入力アドレスの MSB を利用し, 入力するアドレス値に対して左 1 ビットシフトする状態を制御する. 表 4-6 に示したように, 左 1 ビットシフトの真理値表では, CF[1:0] が S₀ (出力データの LSB に埋め込む置換イネーブル) = “1” と T (入力アドレスの MSB を格納) に設定される.

Address		Data[3:0]		CF[1:0]	
Dec	Bin	Dec	Bin	S ₀	T
0	0000	0	0000	1	0
1	0001	2	0010	1	0
2	0010	4	0100	1	0
3	0011	6	0110	1	0
4	0100	8	1000	1	0
5	0101	10	1010	1	0
6	0110	12	1100	1	0
7	0111	14	1110	1	0
8	1000	0	0000	1	1
9	1001	2	0010	1	1
10	1010	4	0100	1	1
11	1011	6	0110	1	1
12	1100	8	1000	1	1
13	1101	10	1010	1	1
14	1110	12	1100	1	1
15	1111	14	1110	1	1

Address		Data[3:0]		CF[1:0]	
Dec	Bin	Dec	Bin	S ₇	T
0	0000	0	0000	1	0
1	0001	0	0000	1	1
2	0010	1	0001	1	0
3	0011	1	0001	1	1
4	0100	2	0010	1	0
5	0101	2	0010	1	1
6	0110	3	0011	1	0
7	0111	3	0011	1	1
8	1000	4	0100	1	0
9	1001	4	0100	1	1
10	1010	5	0101	1	0
11	1011	5	0101	1	1
12	1100	6	0110	1	0
13	1101	6	0110	1	1
14	1110	7	0111	1	0
15	1111	7	0111	1	1

T: 入力アドレスMSB格納
S₀: LSBに埋め込む置換EN

T: 入力アドレスLSB格納
S₇: MSBに埋め込む置換EN

(a) 左 1 ビットシフトの真理値表

(b) 右 1 ビットシフトの真理値表

図 4-14 1 ビットシフトレジスタの真理値表とデータフィールド制御

左 1 ビットシフト時にキャリーしないグループでは, 例えば, “0001” が入力された場合, PMU の入力 COND=“0”, 入力アドレスが “0000” から “0111” までは同様に, Data[3:0] の値が左 1 ビットシフト値として出力される. 次に, 左 1 ビットシフト時にキャリーが発生するグループでは, 例えば “1001” が入力された場合, Data[3:0] の値 “0010” であり, そのまま左 1 ビットシフト値として利用できない. この時, S₀=“1” 設定が有効となり, Data[3:0] の LSB の値に置換され出力される. すなわちデータフィールドの Data[3:0] 値 “0010” の LSB を置換し, “0011” が左 1 ビットシフト値として出力される (PMU のデータ制御部でこの処理を行うが, ここでは説明を省略する).

右 1 ビットシフト時にキャリーが発生しないグループでは, 例えば, “0000” が入力された場合, PMU の入力 COND=“0”, Data[3:0] の値 “0000” が出力される. このようにキャリーしないグループでは, Data[3:0] の値が右 1 ビットシフト値として出力される. 次に, 入力アドレスで右 1 ビットシフト時にキャリーが発生するグループでは, 例えば “0001” が入力された場合, Data[3:0] の値 “0000” が右 1 ビットシフト値として利用できない. この時, S₇ (出力データの MSB に埋め込む置換イネーブル) = “1” 設定が有効となり, Data[3:0] の MSB が置換され出力される. すなわちデータフィールドの Data[3:0] 値 “0000” の MSB を置換し, “1000” が右 1 ビットシフト値として出力される. 以上の仕組みでシフトレジスタ機能を実現している.

図 4-13 の PMU の 1 ビットシフトレジスタモデルに左 1 ビットシフトレジスタの真理値表を実装してシミュレーションを行った結果を図 4-15 に示す。

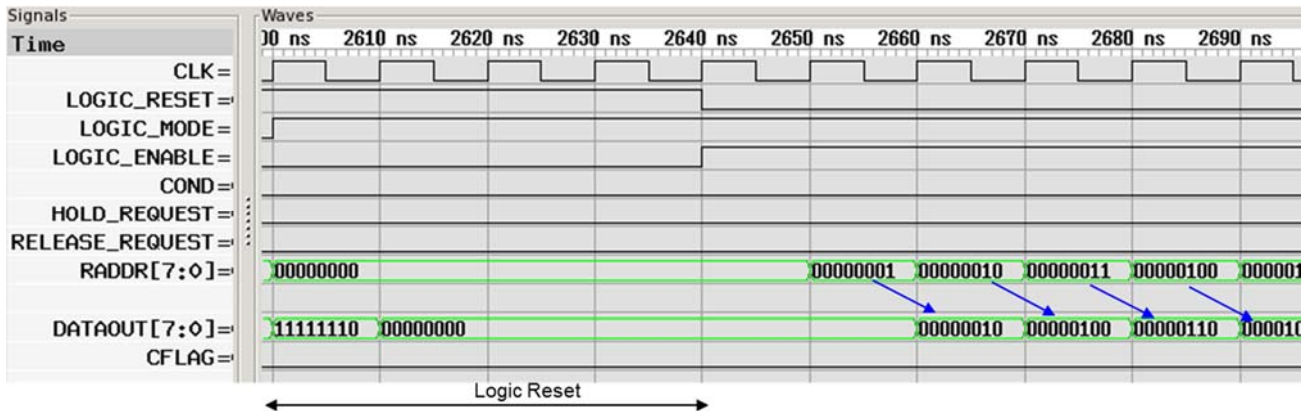


図 4-15 PMU8 ビット左 1 ビットシフトレジスタのシミュレーション波形

入力アドレス RADDR[7:0] = “0000 0001” に対して、DATAOUT[7:0] = “0000 0010” が出力され、以降入力アドレスに対して左 1 ビットシフトした出力値が出力され、問題なく動作していることを確認した。

最終的に表 4-8 に示す 16 種類のシフトレジスタモデルを作成し、8/16/32 ビット長、シフト量 (1 ビット/2 ビット)、および左右シフトとローテートの組み合わせでシミュレーションモデル評価を行った。

表 4-8 PMU のシフトレジスタモデルの種類

シフトレジスタの種類	PMU の数			
	8 ビット		16 ビット	32 ビット
	シフト量			
	1 ビット	2 ビット	1 ビット	1 ビット
左 (論理シフト)	1	2	2	—
左 (ローテート)	1	2	2	—
右 (論理シフト)	1	2	2	—
右 (算術シフト)	1	2	2	4
右 (ローテート)	1	2	2	—

4.2.1.3 演算器モデル

表 4-9 に加算器モデルの動作シミュレーションで利用するマイクロ命令を示す。ここで演算命令が用いられる。PMU での演算では、外部から入力された 8 ビットアドレスの上位 4 ビット、下位 4 ビットをそれぞれ演算対象データとし、その結果を真理値表に準備し、4 ビットの加減算結果を出力する。

アドレスの入力は外部レジスタからとし、データフィールドの真理値表には、キャリー信号入力 (以下、CI:Carry In) が無い場合 (CI=0) の演算結果と CI が有る場合 (CI=1) の双方情報を書き込んでお

く. CI は, PMU の入力 COND の入力信号とし, データ制御部は CI の入力の有無によって出力を選択する.
 CFLAG[1:0]は, CFLAG[1] = 1 で上位 4 ビット, CFLAG[0] = 0 で下位 4 ビットを出力する. 加算器の動作は,

- Addr[7:4], Addr[3:0], CI=0 入力時→CF[0],Data[3:0]
- Addr[7:4], Addr[3:0], CI=1 入力時→CF[1],Data[7:4]

となり, 上位または下位のデータを Dout[3:0]として出力する.

表 4-9 演算器モデルで使用するマイクロ命令

FLAG[7:0]							Data [7:0]							備考	選択アドレス	適用事例		
SEQ		CF		type			CALC _H				CALC _L							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			
ASV	info	0	T _H	T _L	0	1	0	CALC _H				CALC _L			Calc. Function	演算設定	External Address	演算

図 4-16 に 4 ビット加算器のモデルと動作を示す. 入力アドレスの上位 4 ビット, 下位 4 ビットをそれぞれ加算対象のデータと見なし, PMU のアドレス選択は外部レジスタに設定する. SRAM には, CI=1 の場合の CFLAG 出力と演算結果を CF[1]と D[7:4]に, CI=0 の場合は, その CFLAG と演算結果を, CF[0]と D[3:0]にそれぞれ格納する. 例えば, CI=0 の時, 1000 と 0111 の 4 ビット加算する場合, PMU にこれら 4 ビット情報 (a[7:4], b[3:0]) を入力アドレス Addr[7:0] = "1000 0111" とし, かつ CI=0 が入力された場合, フラグフィールドとデータフィールドが読みだされるが, 入力された CI 信号によりデータ制御部で CI=0 の場合の演算結果が選択され, 演算結果 d[3:0]として Data[3:0]=1111 と CF[0]=0 が出力される.

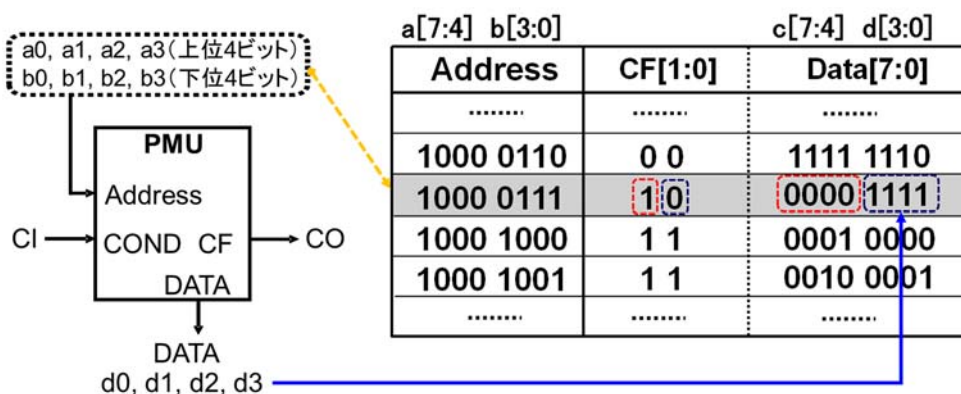


図 4-16 PMU の 4 ビット加算器のモデルと動作

ここで, 出力された CF[1:0]は, キャリー信号として他の PMU の入力 COND に接続することで演算ビット幅を拡張することができる. PMU を 2 個カスケード接続して 8 ビット加算器を構成した例を図 4-17 に示す.

この場合, 加算する値, すなわち入力アドレスが A = (a₇, a₆, ..., a₀) および B = (b₇, b₆, ..., b₀) とすると, これらアドレスを各下位 4 ビットに連結したものを 1 段目の PMU に, 各上位 4 ビットを連結したものを 2 段目の PMU にアドレスとして与える. 1 段目の PMU には CI=0 を与え, 各データ制御部から

出力された 2 段目の 4 ビットを上位 4 ビットとして、1 段目の 4 ビットを下位 4 ビットとして連結することで 8 ビットの演算結果が得られる。また、1 段目の PMU に CI=1 に設定することで減算結果が得られる。この加算方法について 8 ビット加算器モデルで説明する。

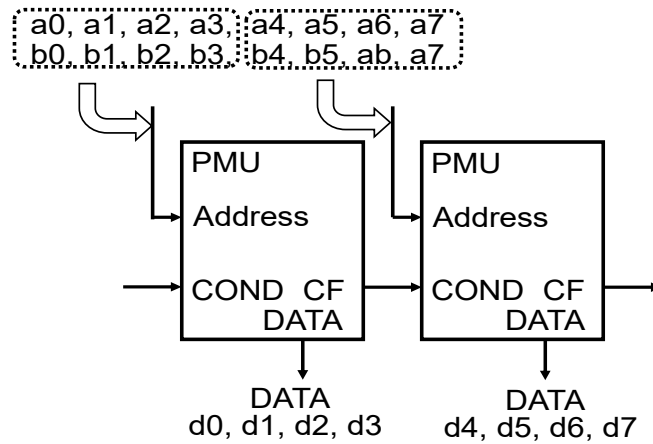


図 4-17 PMU による 8 ビット加算器モデル

PMU の加算方法 (例, $A + B = C$)

1) アドレス A = 0x81 (1000 0001)

アドレス B = 0x7F (0111 1111)

4 ビットに分割して 1 段目の PMU に入力する。

下位 4 ビットアドレス加算では, (0001 1111) が入力アドレスとなり,

この時の出力値は, Cond=0, すなわち CI=0 となり, Data[3:0] が選択され,

(0001 0000) CFLAG=1 となる。

2) 同じく, 4 ビットに分割して 2 段目の PMU に入力する上位 4 ビットアドレス加算では,

(1000 0111) となる。

この時の出力値は, 1 段目の PMU の CFLAG=1 が 2 段目の PMU の Cond に入力され, Cond=1,

すなわち CI=1 となり, Data[7:4] が選択され,

(0000 1111) CFLAG=1 となる。

最終的に上位・下位の青字下線部が出力値として選択され (0000 0000) の Carry=1 となる。

したがって, アドレス A (1000 0001) + アドレス B (0111 1111) = C (0000 0000) CFLAG=1

となる。

図 4-18 に 8 ビット加算器 PMU のモデルを示す。また, 図 4-17 に上記加算方法の例に示したシミュレーション結果を示す。設計通りの計算結果が出力され, 問題なく動作していることを確認した。この加算シミュレーション波形では, この時の HEX 値が表示されている。また, この方法で, PMU を 4 個接続し, 16 ビットの加算器も評価した。図 4-20 にそのシミュレーションモデルの構成と結線を示す。

以上により, 表 4-10 に示す 3 種類の演算器モデルを作成し, 評価を行った。8/16 ビット長の加算器および 8 ビットの減算器でシミュレーションモデルを作成し評価を行った。

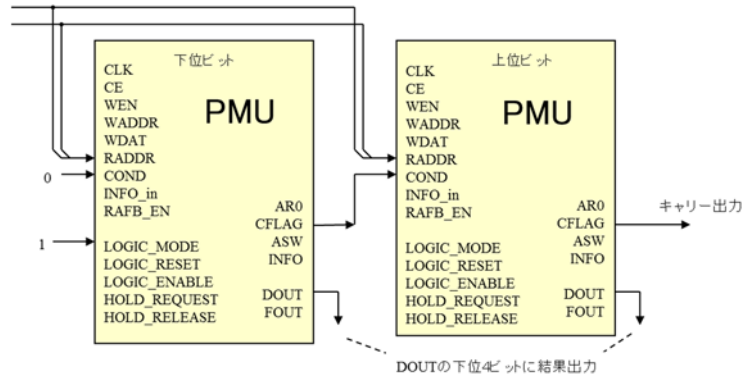


図 4-18 PMU の 8 ビット加算器シミュレーションモデルと結線

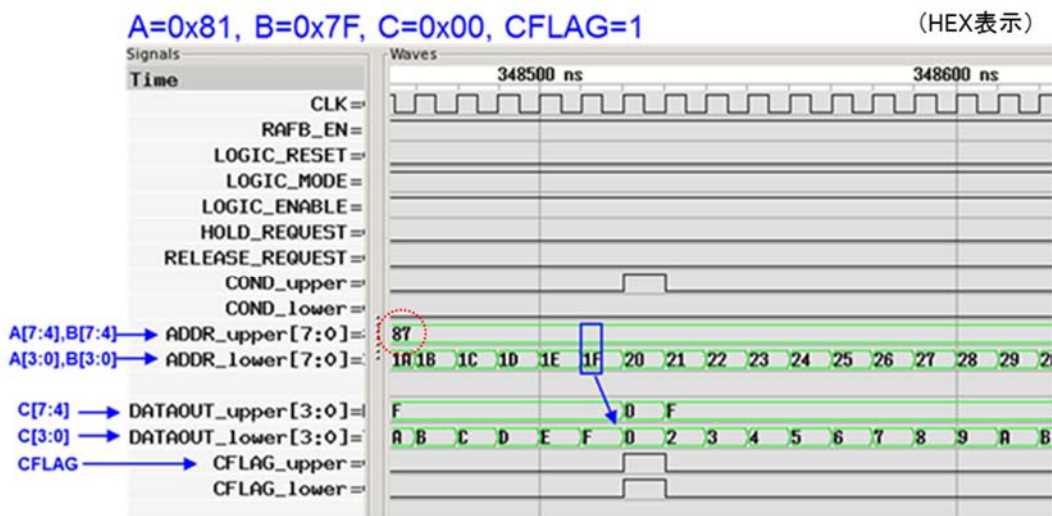


図 4-19 8 ビット加算器のシミュレーション結果

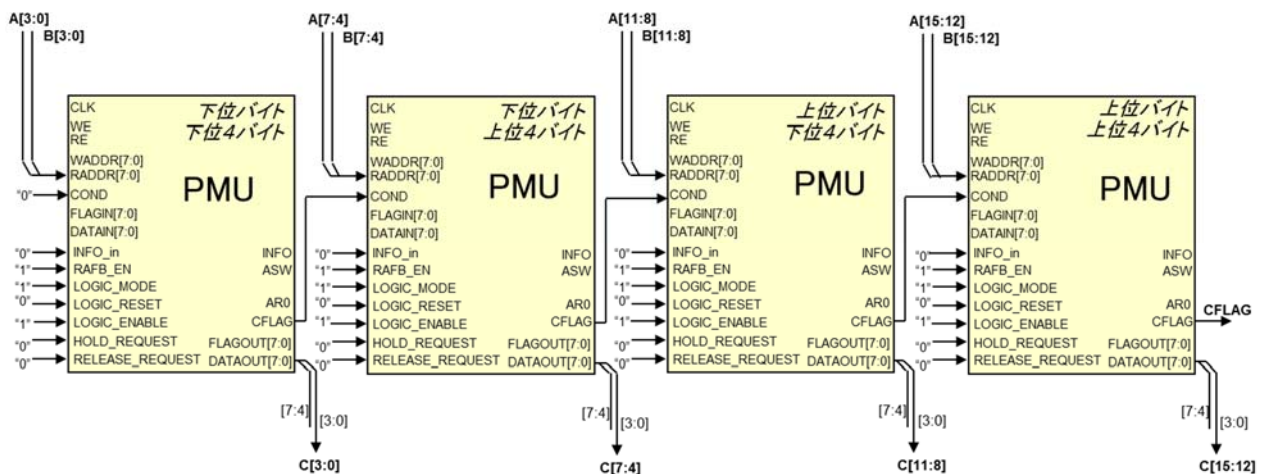


図 4-20 PMU の 16 ビット加算器シミュレーションモデルと結線

表 4-10 PMU の演算器モデルの種類

演算器	PMU の数	
	8 ビット	16 ビット
加算器	2	4
減算器	2	-

4.2.2 マイコン周辺回路のモデリングと評価

汎用的なマイコンに搭載される周辺回路として FIFO，シリアル通信インタフェース（Serial Communication Interface : SCI）および PWM を選びモデル評価を行った。また，この中から 8 ビット PWM を選び FPGA 上に実装・評価を行い，動作確認・波形観測を行った結果を述べる。

4.2.2.1 FIFO モデル

一般に FIFO は，データ通信などのデータ送受信のバッファとしてよく利用される（図 4-21）。様々な実装方法があるが，ここでは Full と Empty だけで制御するポインタ型 FIFO を前提とする。

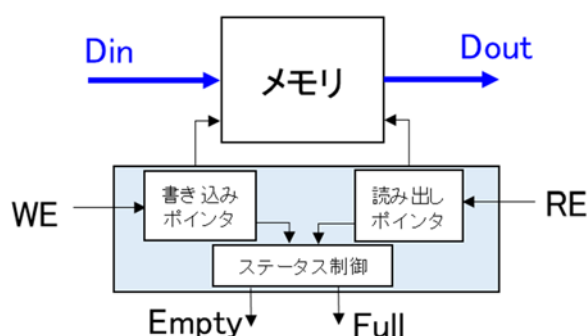


図 4-21 一般的な FIFO の構成

基本動作は，

- 1) 入力 Din に対して，FIFO のバッファメモリが満杯の場合，full 信号を “High”，出力 Dout に対しては，FIFO のバッファメモリが空いている場合，empty 信号 “High” を発行する。
- 2) 書き込み側では，full 信号が “high” の状態では書き込みはできない。full 信号 “Low” 状態まで待って，書き込みイネーブル信号 WE を使って書き込む。
- 3) 読み出し側では，Empty 信号が “high” の状態ではデータが空のため読み出しはできない。Empty 信号 “Low” 状態まで待って，読み出しイネーブル信号 RE を使って読み出す。

この Write Enable (WE) /Read Enable (RE) は書き込みおよび読み込みポインタとしてステータス制御部で管理される。この書き込みおよび読み込みポインタ部は PMU のインクリメンタが利用され，さらに PMU を使ってバッファメモリ，カウンタ機能を実装して FIFO 機能の評価を行った。

表 4-11 に FIFO モデルの動作シミュレーションで利用するマイクロ命令を示す。書き込みポインタおよび読み込みポインタ用の条件付きインクリメント，スルー機能のマイクロ命令が用いられる。また，

FIFO では制限のあるバッファサイズの読み出し/書き込み制御を行う必要があり、ここでは8ビット長の深さ16ワードとし、ステータスとして Full, Empty フラグをサポートする。

図 4-22 に 16 ワードの FIFO シミュレーションモデルとその結線を示す。このモデルでは、PMU①は書き込み、読み込みのポインタとステータス制御を行う。図 4-23 にその動作を示す。

表 4-11 FIFO モデルで使用するマイクロ命令

FLAG[7:0]						Data [7:0]						備考	選択アドレス	適用事例			
SEQ		CF		type													
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0		
FH	FL	1	T _H	T _L	0	1		COUNT _H			COUNT _L			4bit counter	条件付インクリメント	Address Reg. +1,+16	FiFo(下位4ビット)
FH	FL	FC	T _H	T _L	0	1		COUNT _H			COUNT _L				条件付デクリメント		FiFo(上位4ビット)
Ctrl Info.						0	T	1	0	0	Control Info.			Ex.t. Jump	外部無条件分岐	External Address	スルー

PMU①では、書き込み、読み込み独立にインクリメント制御を行い、ポインタを管理すると同時にメモリのフラグを使ってステータスを発行する。さらに FIFO ポインタを次段の PMU②に発行する。このポインタ情報は、データ出力の下位4ビットを FIFO メモリのライトアドレス、データ出力の上位4ビットを FIFO のリードアドレスに設定してある。

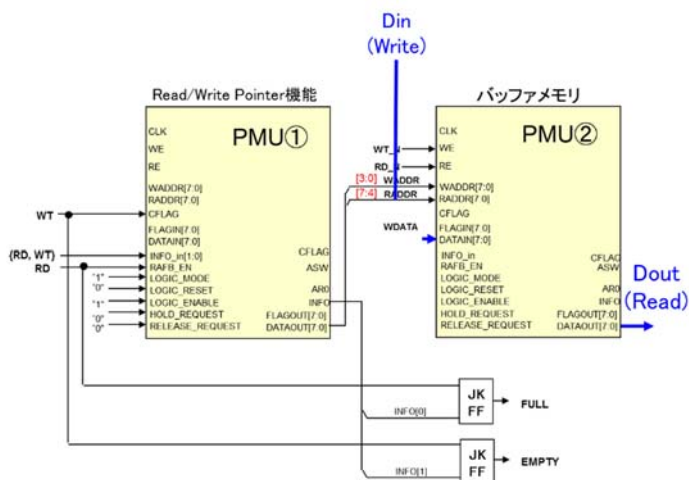


図 4-22 16 ワードの深さの FIFO シミュレーションモデルと結線

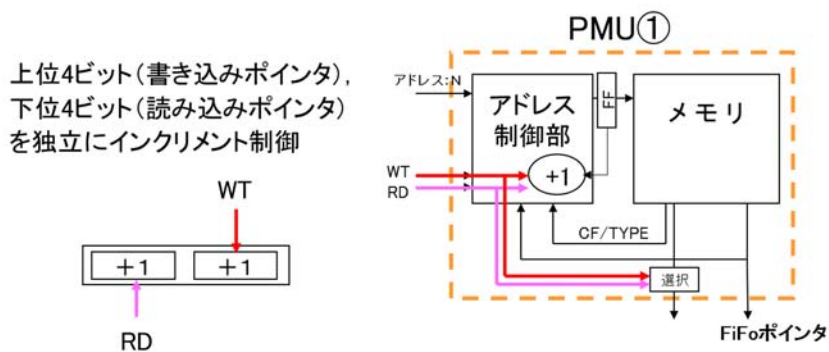


図 4-23 PMU①の書き込み/読み込みポインタ制御

以上のような環境で FIFO の書き込み，読み込みと Full/Empty 信号の動作をシミュレーションで確認した．その一例を図 4-24 に示す．ここでは，16 回の書き込みで Full 信号が，16 回の読み込みで Empty が観測された．

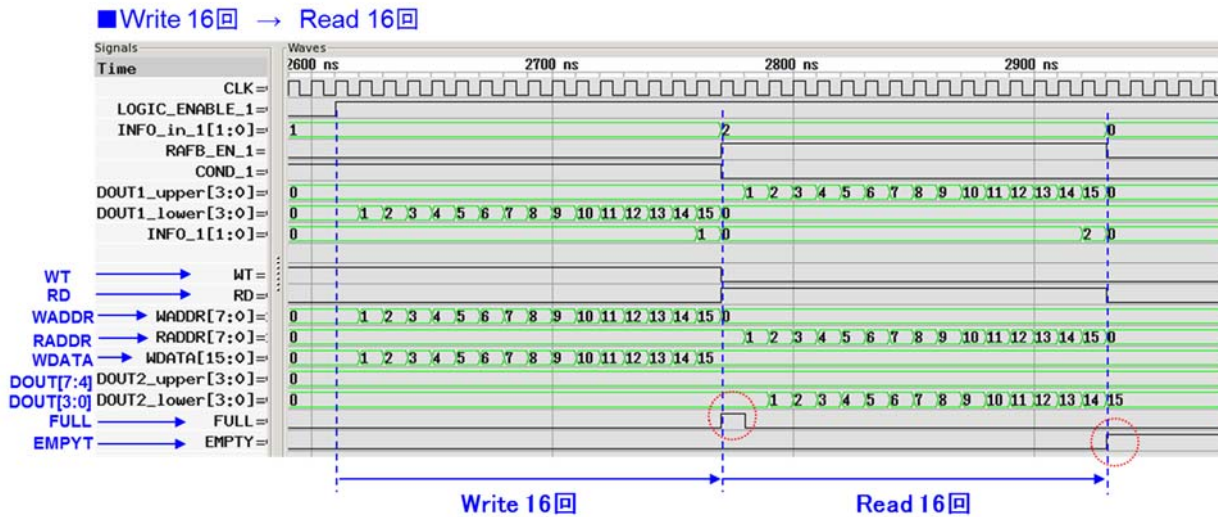


図 4-24 FIFO の Full/Empty 信号の確認

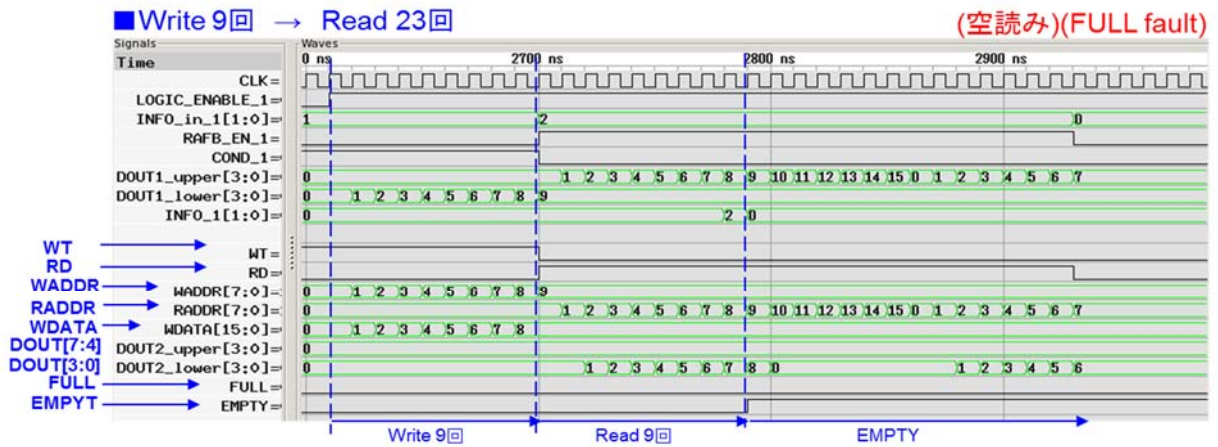


図 4-25 FIFO の Read/Write 評価

次にその他の書き込み/読み込み条件でのシミュレーション結果を図 4-25 に示す。

上段は、9 回書き込みを行った後、読み込みを 23 回実行した結果である。9 回書き込みを行い、23 回の読み込み中に、9 回目の読み込み実行後、Empty 信号が“High”に立ち上がるのが確認でき、9 回目以降は空読み状態となり、Full error 状態となる。

また、下段は 20 回の書き込みと 16 回の読み込みを行った結果である。16 ワードのバッファしかないため、書き込み 16 回目以降は上書き状態となり、16 回目以降 20 回目まで Full 信号が“High”となり、書き込み終了後、読み込みが上書き分の影響で、Empty error 状態となる。以上のように Empty 信号が“High”状態で、読み込みイネーブル信号“High”，または Full 信号が“High”状態で、書き込みイネーブル信号“High”，はルール違反となる。これら書き込み/読み込みを組み合わせた 33 パターンのシミュレーション検証を行ったが、いくつかのパターンは、動作保証しないか、読めないよう、また書き込めないようにブロック制御する必要がある。ここでは、これらの制御は今後の製品化時の課題とした。

4.2.2.2 シリアル通信インタフェースモデル

シリアル通信のための SCI には非同期型、同期型があるが、ここでは同期型シリアル通信インタフェースを前提に実装を行った。図 4-26 にシリアル通信の概要を示す。

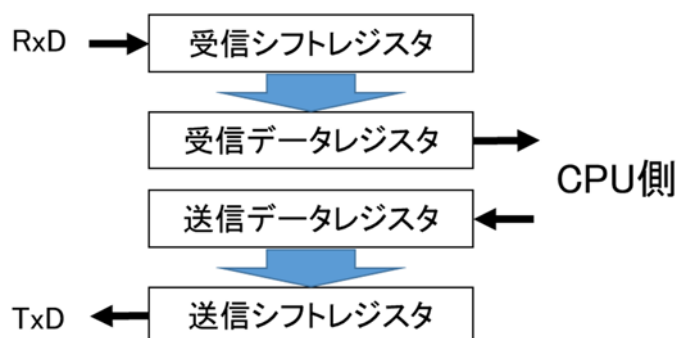


図 4-26 シリアル通信の概要

基本動作は、

- 1) データの受信時、RxD 信号が LSB→MSB の順に受信シフトレジスタ 1 ビット毎に格納される。8 ビット分が全て正常に受信された時点で受信データレジスタに 1 バイト分記憶される。CPU には受信完了が通知され、受信データレジスタ記憶された 1 バイトのデータを読み出す。
- 2) データ送信時、送信シフトレジスタが空、すなわちデータ送信完了になると送信データレジスタに新たに書き込まれたデータを送信シフトレジスタに書き込み、データの送信 (TxD) を始める。
このように、通常のレジスタとシフトレジスタを利用したシンプルな構成になっている。これに送受信制御を加えて、PMU を用いたシリアル通信インタフェースのモデルを構成した。

機能仕様は、以下とした。

【PMU のシリアル通信インタフェース仕様】

- (1) クロックは外部から供給 (ポーレートジェネレータは搭載しない)

- (2) クロックのデューティは 50%
- (3) フレームフォーマットは、8 ビットデータ固定長（パリティ、CRC はサポートしない）
- (4) LSB ファーストで送受信
- (5) 1 フレーム受信完了/送信完了ステータス発行
- (6) 受信オーバーランエラー検出しない
- (7) 送信データエンプティステータスなし
- (8) 送信終了ステータス（ソフト対応）

【受信側モデル】

ここでは (1) ~ (5) の要件に対応するモデル構築を行った。また、モデル実装では受信側と送信側を分割して実装とした。図 4-27 に PMU を使ったシリアル通信受信側のシミュレーションモデルとその結線を示す。

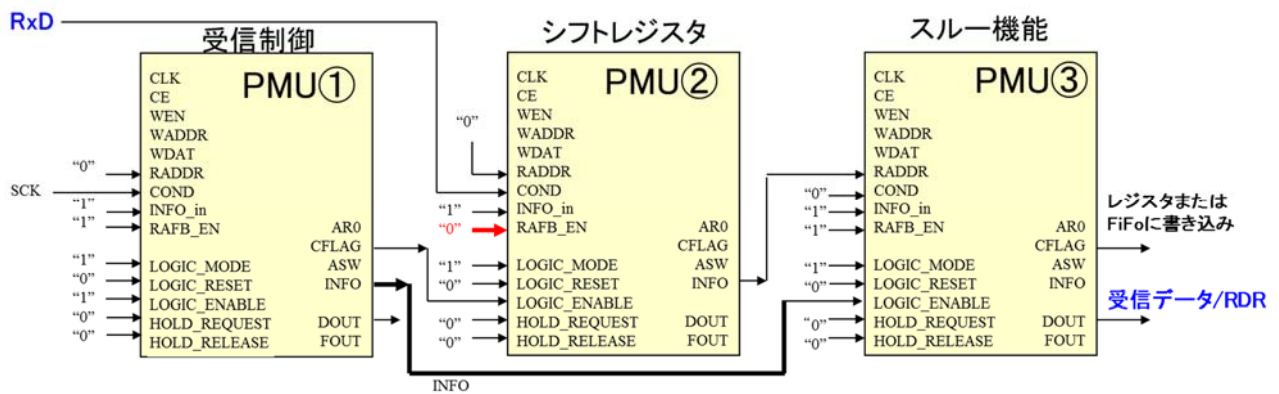


図 4-27 シリアル通信（受信側）シミュレーションモデルとその結線

このモデルは、PMU①はデータの受信シーケンスの制御機能、PMU②はシリアルデータを受信する左シフトレジスタ、PMU③はスルー機能を搭載し構成されている。PMU①は Serial Clock (SCK) を監視し、次段に動作タイミング信号を出力する。PMU②は受信用のシフトレジスタでPMU①のCFLAGが LOGIC_ENABLE に入力され動作する。PMU③は受信データレジスタとして機能する。PMU②の INFO 信号をアドレスとして入力、スルー機能で 8 ビット固定長のデータを出力し、連続的に受信データを出力する動作を行う。図 4-28 にこのモデルの受信動作を示す。クロック同期式で、LSB ファーストで 8 ビットデータ (D0~D7) を受信する。

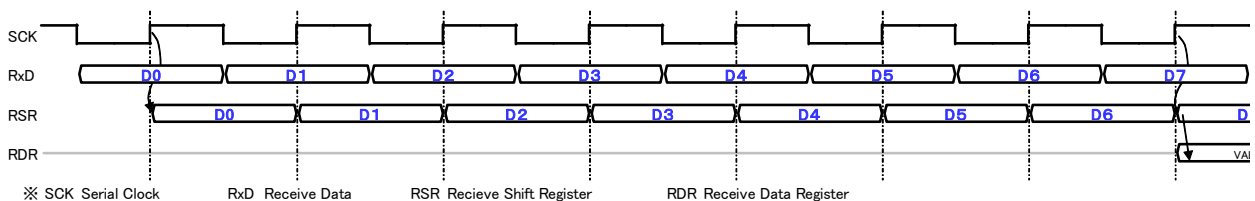


図 4-28 シリアル通信（受信側）モデルの受信動作

次にこの受信制御のシーケンス動作について述べる．図 4-29 に受信動作フローチャートを示す．

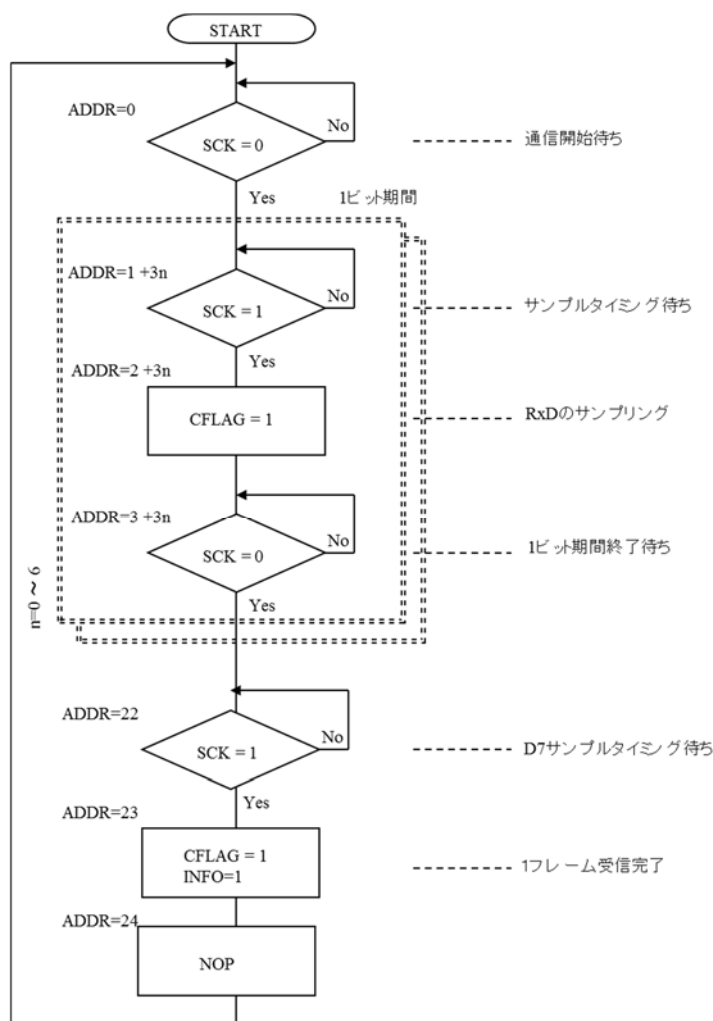


図 4-29 受信動作のフローチャート

このフローチャートは受信制御シーケンス動作を表したものであるが，これを PMU①に実装すると図 4-30 の真理値表に翻訳される．以下，この真理値表をベースに受信動作フローを説明する．説明の都合上この真理値表の入力アドレスを十進数で表現（“0000 0111” = 7）する．この真理値表では図 4-29 で示したフローチャートがアドレス 0 から 24 までの 25 ステップ（サイクル数）で表現されている．

最初の入力アドレス 0 では，通信待ち状態となっている．これ以降は，各データ毎（D0～D6）で図 4-29 に示した①サンプルタイミング待ち（SCK=1 の立ち上がりタイミング），②RxD サンプリング

（CFLAG=1 の立ち上がりタイミング），③SCK の 1 ビット期間終了待ちのいずれかの条件分岐判定が行われる．アドレス 1～3 までは D0 の判定，アドレス 4～6 までは D1 の判定と D6 のアドレス 19～21 まで実行され，最後の D7（アドレス 22～23）では，タイミング待ち/受信完了判定が行われ，INFO と CFLAG の状態で受信完了が判定されるとアドレス 24 に遷移して，受信データレジスタをリセットし，受信待ち状態のアドレス 0 に移行する．

受信制御シーケンス

Function : Clocked Serial Receive Control

Input			Output								
Address			Data			Reserved		CF		TYPE	
DEC	HEX	BIN	DEC	HEX	BIN	INFO	ASW	1	0		
0	00	00000000	0	00	00000000	00	0	1	0	101	COND=1 なら自分 COND=0 待ち
1	01	00000001	1	01	00000001	00	0	1	0	100	COND=0 なら自分 COND=1 待ち
2	02	00000010	3	03	00000011	00	0	0	1	000	D0
3	03	00000011	3	03	00000011	00	0	1	0	101	
4	04	00000100	4	04	00000100	00	0	1	0	100	D1
5	05	00000101	6	06	00000110	00	0	0	1	000	
6	06	00000110	6	06	00000110	00	0	1	0	101	COND=0 なら自分 COND=1 待ち
7	07	00000111	7	07	00000111	00	0	1	0	100	D2
8	08	00001000	9	09	00001001	00	0	0	1	000	
9	09	00001001	9	09	00001001	00	0	1	0	101	COND=0 なら自分 COND=1 待ち
10	0A	00001010	10	0A	00001010	00	0	1	0	100	D3
11	0B	00001011	12	0C	00001100	00	0	0	1	000	
12	0C	00001100	12	0C	00001100	00	0	1	0	101	COND=0 なら自分 COND=1 待ち
13	0D	00001101	13	0D	00001101	00	0	1	0	100	D4
14	0E	00001110	15	0F	00001111	00	0	0	1	000	
15	0F	00001111	15	0F	00001111	00	0	1	0	101	COND=0 なら自分 COND=1 待ち
16	10	00010000	16	10	00010000	00	0	1	0	100	D5
17	11	00010001	18	12	00010010	00	0	0	1	000	
18	12	00010010	18	12	00010010	00	0	1	0	101	COND=0 なら自分 COND=1 待ち
19	13	00010011	19	13	00010011	00	0	1	0	100	D6
20	14	00010100	21	15	00010101	00	0	0	1	000	
21	15	00010101	21	15	00010101	00	0	1	0	101	COND=0 なら自分 COND=1 待ち
22	16	00010110	22	16	00010110	00	0	1	0	100	D7
23	17	00010111	24	18	00011000	01	0	0	1	000	
24	18	00011000	0	00	00000000	00	0	0	0	000	COND=0 なら自分 COND=1 待ち
25	19	00011001	0	00	00000000	00	0	0	0	000	受信完了 RDRセット

受信完了ステータス用のトリガ

図 4-30 受信制御シーケンスの真理値表と受信完了トリガ

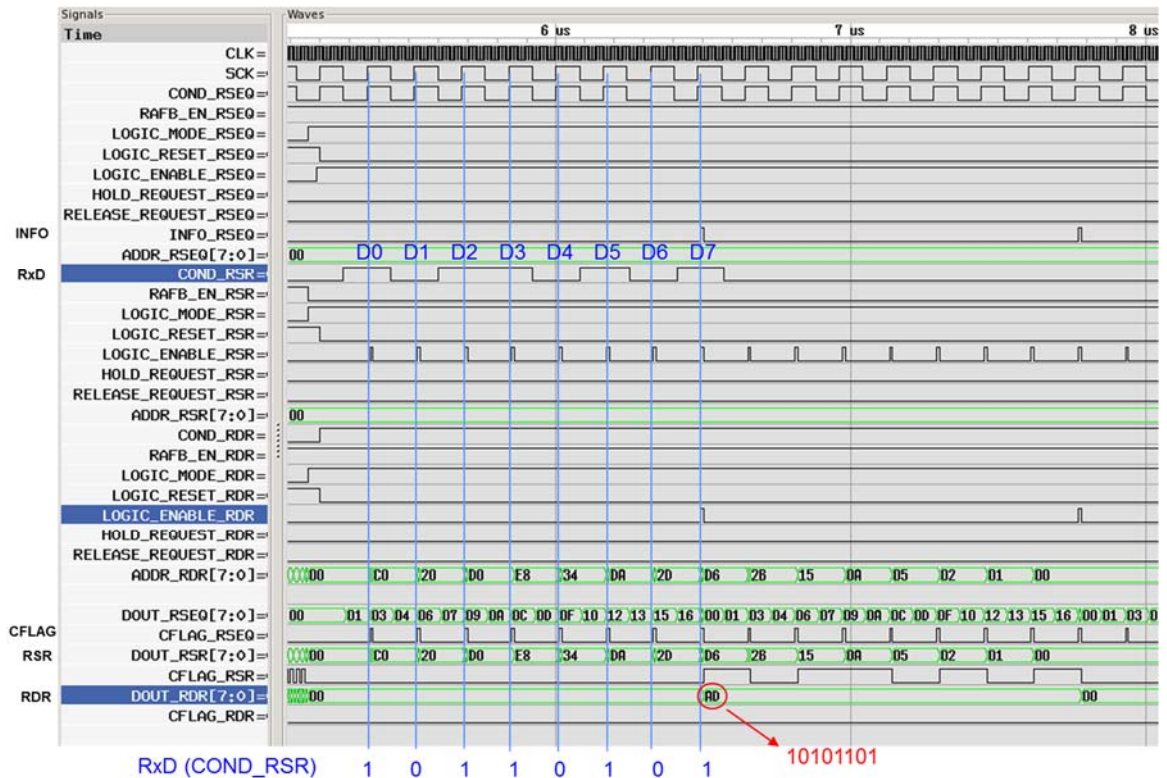


図 4-31 シリアル通信（受信側）シミュレーション波形

以上のようなモデル構成に、動作（サブルーチン）実装を行い、シリアル通信の受信シミュレーションを行った結果を図 4-31 に示す。RxD から試験用の受信データ “1011 0101” を入力すると、受信シフトレジスタはデータを受信、LOGIC_ENABLE が “High” のタイミングでデータを確認すると D0～D7 の順に “1011 0101” を受信していることがわかる。これは LSB ファーストで受信した信号である。これを受信データレジスタ（RDR）に転送時に、データは MSB に変換され、RDR に “AD (Hex) = 1010 1101 (Bin)” が記憶され、問題なく動作していることを確認した。

【送信モデル】

図 4-32 に PMU を使ったシリアル通信送信側シミュレーションモデルとその結線を示す。

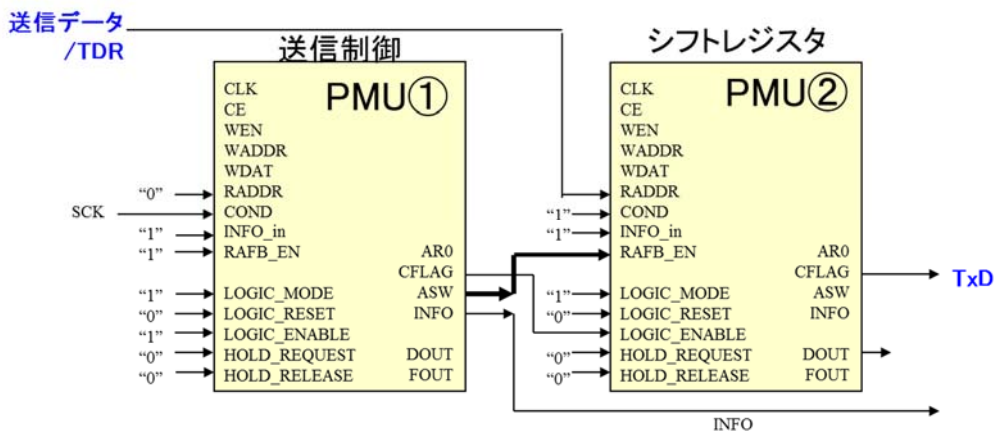


図 4-32 シリアル通信（送信側）シミュレーションモデルとその結線

受信モデルと同様に、PMU①はデータの送信シーケンスの制御機能、PMU②はシリアルデータを送信する左シフトレジスタで構成されている。PMU①は SCK を監視し、次段に動作タイミング信号を出力する。PMU②は送信用のシフトレジスタで PMU①の CFLAG が LOGIC_ENABLE に入力され、1 ビット毎に送信を行う。図 4-33 にこのモデルの送信動作を示す。受信側と同様にクロック同期式で、LSB ファーストで 8 ビットデータ（D0～D7）を送信する。

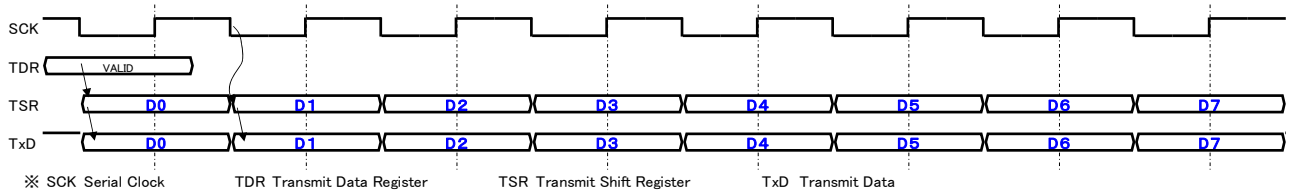


図 4-33 シリアル通信（送信側）モデルの送信動作

次にこの送信制御のシーケンス動作について説明する。図 4-34 に送信動作フローチャートを示す。このフローチャートは送信制御シーケンス動作を表したものであるが、これを PMU①に実装すると図 4-35 の真理値表に翻訳される。受信側と同様に、この真理値表をベースに送信動作フローを説明する。

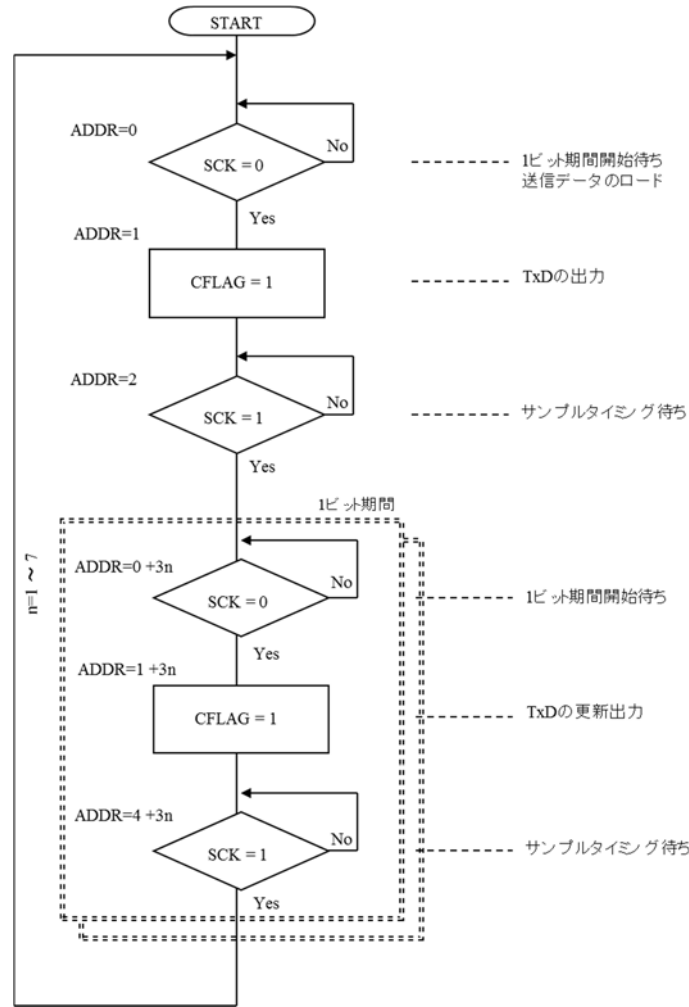


図 4-34 送信動作のフローチャート

説明の都合上この真理値表の入力アドレスを十進数で表現 (“0000 0111” = 7) する。この真理値表では図 4-34 で示したフローチャートがアドレス 0 から 24 までの 25 ステップで表現されている。

送信時では、SCK を監視し、SCK の立下りのタイミングによって、アドレス 0/アドレス 2 またはアドレス 1 (CFLAG) の状態を判定して、送信開始時の PMU②のシフトレジスタのアドレス値 (D0) を決める。これ以降、各データ毎 (D1~D7) で図 4-35 に示した SCK の①1 ビット期間開始待ち (SCK=0)、② TxD の更新 (CFLAG=1 の立ち下がりタイミング)、③サンプルタイミング待ち (SCK=1 の立ち下がりタイミング) のいずれかの条件分岐判定が行われる。アドレス 0~2 までは D0 の判定、アドレス 3~5 までは D1 の判定と D7 のアドレス 21~23 まで実行され、最後のアドレス 24 では、INFO[0]=1 となり送信が完了し、再びアドレス 0 に移行する。

以上のようなモデル構成に動作 (サブルーチン) 実装を行い、シリアル通信の送信シミュレーションを行った結果を図 4-36 に示す。送信シフトレジスタ (TDR) に試験用送信データ “0011 0011” を入力する。LOGIC_ENABLE が “High” のタイミングで、TxD を観測すると送信データ D0~D7 の順に “1100 1100” を送信していることがわかる。これは LSB ファーストで送信した信号であり、問題なく動作していることを確認した。

送信制御シーケンス

Function : Clocked Serial Transmit Control

Input			Output							
Address			Data			Reserved		CF		TYPE
DEC	HEX	BIN	DEC	HEX	BIN	INFO	ASW	1	0	
0	00	00000000	0	00	00000000	00	1	1	0	101
1	01	00000001	2	02	00000010	00	1	0	1	000
2	02	00000010	2	02	00000010	00	0	1	0	100
3	03	00000011	3	03	00000011	00	0	1	0	101
4	04	00000100	5	05	00000101	00	0	0	1	000
5	05	00000101	5	05	00000101	00	0	1	0	100
6	06	00000110	6	06	00000110	00	0	1	0	101
7	07	00000111	8	08	00001000	00	0	0	1	000
8	08	00001000	8	08	00001000	00	0	1	0	100
9	09	00001001	9	09	00001001	00	0	1	0	101
10	0A	00001010	11	0B	00001011	00	0	0	1	000
11	0B	00001011	11	0B	00001011	00	0	1	0	100
12	0C	00001100	12	0C	00001100	00	0	1	0	101
13	0D	00001101	14	0E	00001110	00	0	0	1	000
14	0E	00001110	14	0E	00001110	00	0	1	0	100
15	0F	00001111	15	0F	00001111	00	0	1	0	101
16	10	00010000	17	11	00010001	00	0	0	1	000
17	11	00010001	17	11	00010001	00	0	1	0	100
18	12	00010010	18	12	00010010	00	0	1	0	101
19	13	00010011	20	14	00010100	00	0	0	1	000
20	14	00010100	20	14	00010100	00	0	1	0	100
21	15	00010101	21	15	00010101	00	0	1	0	101
22	16	00010110	23	17	00010111	00	0	0	1	000
23	17	00010111	23	17	00010111	00	0	1	0	100
24	18	00011000	0	00	00000000	00	0	0	0	000
25	19	00011001	0	00	00000000	00	0	0	0	000

送信用シフトレジスタのアドレス入力切替
送信データFIFOの取り込みのトリガとして利用可能

COND=1 なら自分 COND=0 待ち

D0 COND=0 なら自分 COND=1 待ち
COND=1 なら自分 COND=0 待ち

D1 COND=0 なら自分 COND=1 待ち
COND=1 なら自分 COND=0 待ち

D2 COND=0 なら自分 COND=1 待ち
COND=1 なら自分 COND=0 待ち

D3 COND=0 なら自分 COND=1 待ち
COND=1 なら自分 COND=0 待ち

D4 COND=0 なら自分 COND=1 待ち
COND=1 なら自分 COND=0 待ち

D5 COND=0 なら自分 COND=1 待ち
COND=1 なら自分 COND=0 待ち

D6 COND=0 なら自分 COND=1 待ち
COND=1 なら自分 COND=0 待ち

D7 COND=0 なら自分 COND=1 待ち

送信完了

送信完了ステータス用のトリガ

図 4-35 送信制御シーケンスの真理値表と送信完了トリガ

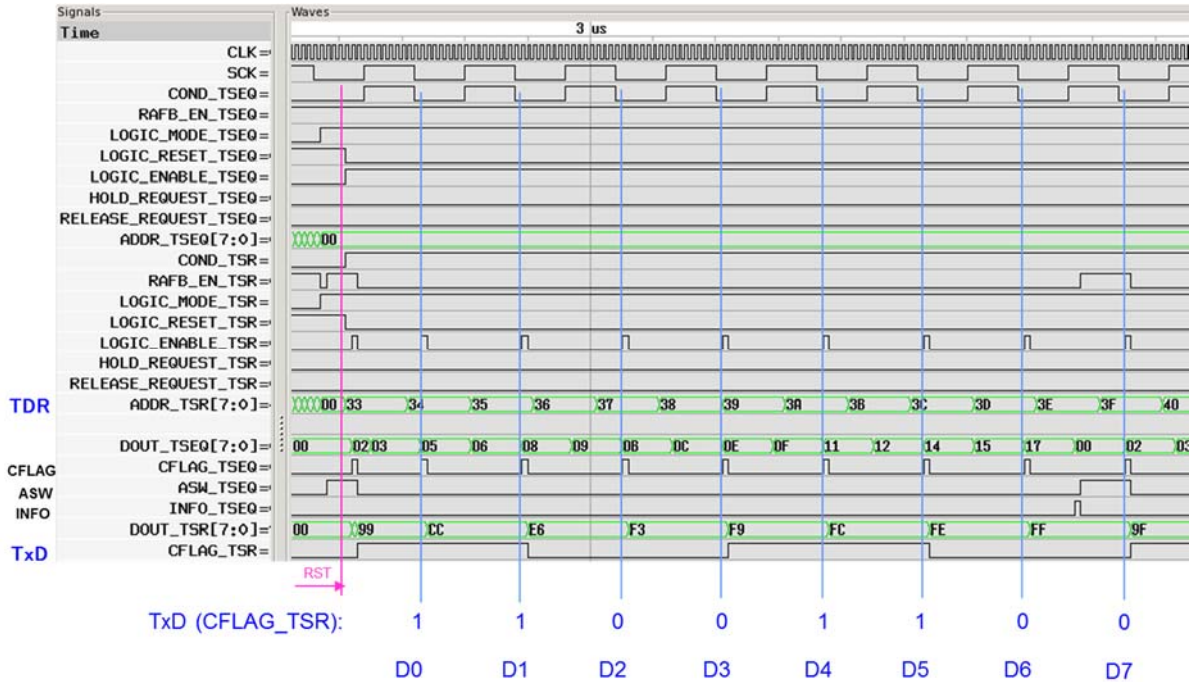


図 4-36 シリアル通信（送信側）シミュレーション波形

4.2.2.3 PWM モデル

PWM は、図 4-37 で示したように、1 波長の周期のパルス幅を自由に変更する事ができる。ここで、パルス幅 tpw と周期 T の比で PWM 信号のデューティ比 D (%) が定義される。

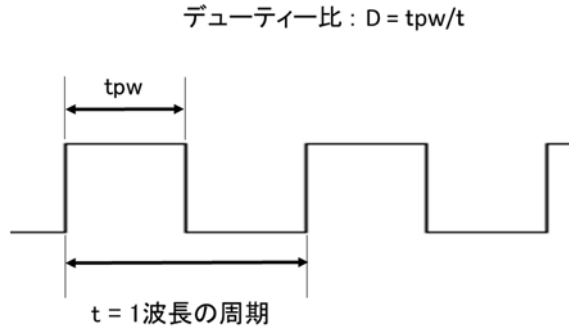


図 4-37 PWM 波形

マイコンでは一般的に、カウンタ/タイマを使った波形生成が用いられる。このような場合には、マイコンのクロックを分周して 1 波長の周期（以下、 T ）を生成し、レジスタ等にその分周値のカウント値を設定することでデューティ比 D を実現している。その一例を図 4-38 に示す。

このように、PWM の制御精度を作るクロックの分周機能、PWM の周期決めるカウンタ機能およびデューティ比を決めるカウンタ機能があれば PWM 機能を実現できる。そこで、3 個の PMU を使って 8 ビット精度の PWM のシミュレーションモデルを構成した。8 ビット PWM は、3 個の PMU をそれぞれ分解精度設定部（分周器）、周期設定部（ダウンカウンタ）およびパルス幅制御部（ダウンカウンタ）の 3 つの機能部品として用いる。図 4-39 に 8 ビット PWM のシミュレーションモデルの構成と結線モデル、および各 PMU に実装する真理値表の一部を示す。PMU①には 8 ビット分周機能、PMU②および PMU③にはダウンカウンタ機能が実装され、各設定用レジスタが準備されている。

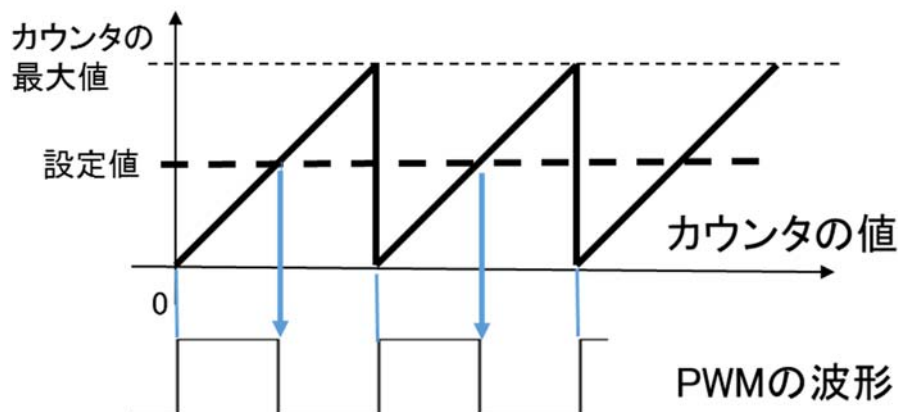


図 4-38 カウンタと PWM 出力波形

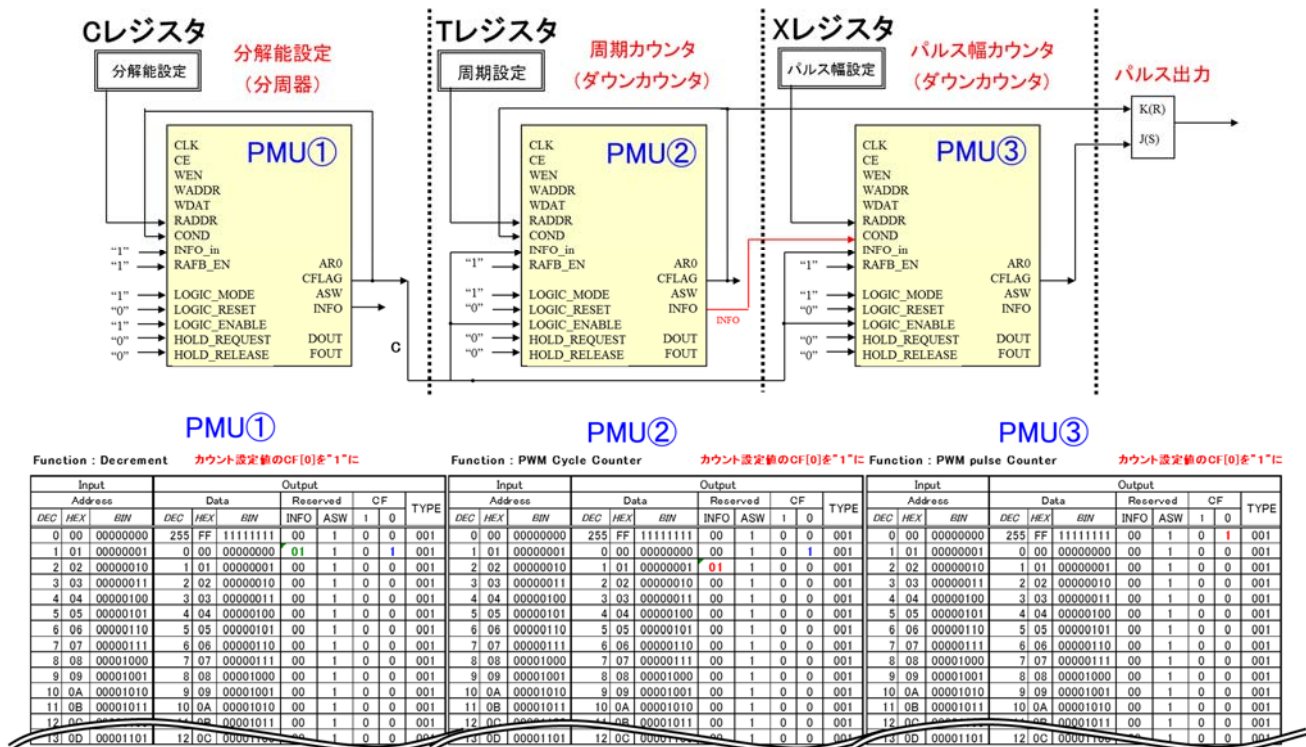


図 4-39 8 ビット PWM のモデル構成と結線および各 PMU の真理値表

第 1 段目の PMU①は、システムクロックを 8 ビット（256）で分周し、PWM のパルス幅の制御ステップ精度を設定する。設定は、C レジスタで行う。次に第 2 段目の PMU②は周期 T を第 1 段目の PMU①が生成する分周した値をカウントして周期 t を生成する。このカウント値の設定は T レジスタで行う。最後に第 3 段目の PMU③は、PMU②と同様に第 1 段目の PMU①が生成する分周した値をカウントしてパルス幅（またはデューティ比 D）を決定する。このカウント値の設定は X レジスタで行う。

具体的には、PMU①に供給されたシステムクロック（CLK）の周波数は分解能 8 ビット（256）で分割される。

この時の分解精度 a は

$$a = (1/CLK) * C \quad (a: \text{分解精度}, CLK: \text{システムクロック}, C: \text{分周設定値/C レジスタ設定値})$$

で表される。ここで、分解精度 a は PMU①の C レジスタで設定され、CFLAG1 として出力される。

次に PMU②は PMU①分周の値を使って周期 t を決定する。

$$t = a * T \quad (t: \text{パルスの周期}, a: \text{分解精度}, T: \text{周期設定値/T レジスタ設定値})$$

で表される。ここで周期設定値 T は T レジスタで設定され、CFLAG2 として出力される。

最後に PMU③では PMU①分周の値を使ってパルス幅 w（デューティ比）を決定する。

$$w = a * X \quad (w: \text{パルス幅}, a: \text{分解精度}, X: \text{パルス幅設定値/X レジスタ設定値})$$

で表される。ここでパルス幅設定値 X は X レジスタで設定され、CFLAG3 として出力され、パルス幅の Low 期間を定義する。以上により JK-FF から指定したパルス幅の波形が出力される。図 4-40 に PMU を用いた 8 ビット PWM の構成とシミュレーション波形の例を示す。

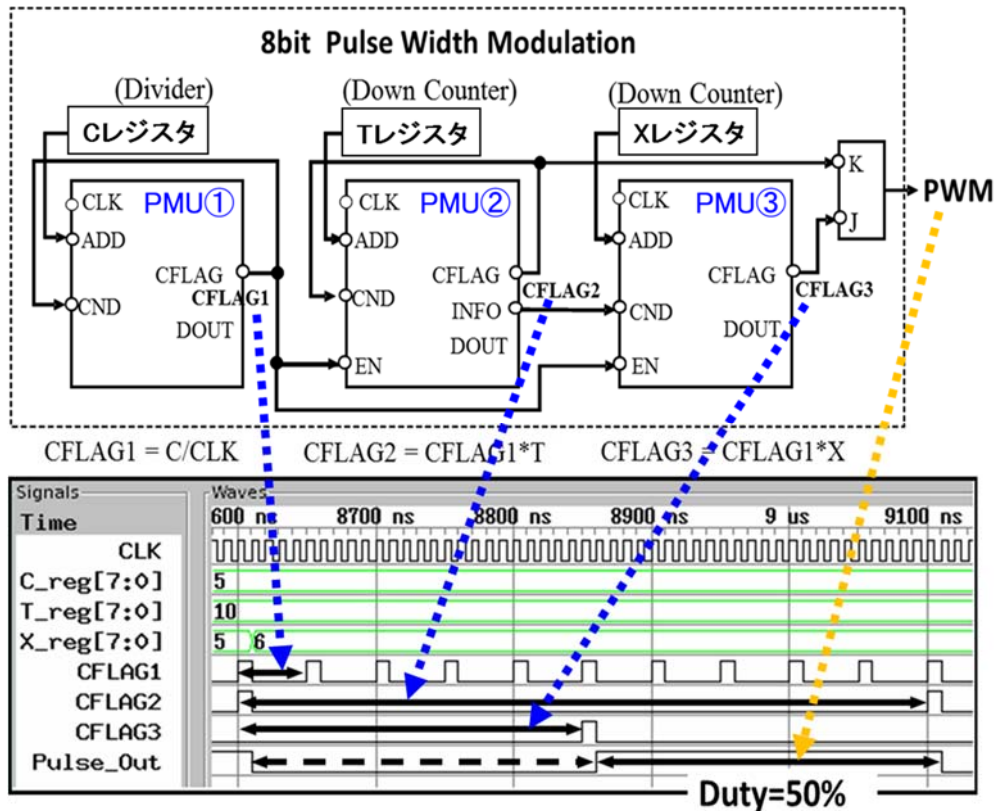


図 4-40 PMU を用いた 8 ビット PWM の構成とシミュレーション波形

ここでは、C レジスタ=5、T レジスタ=10、X レジスタ=5 が設定されたシミュレーション波形になっている。PMU①はシステムクロックを 5 分周した分解能の信号 CFLAG1 を出力し、次に PMU②はこの CFLAG1 を 10 回カウント後、周期 CFLAG2 を出力する。最後に PMU③は指定されたパルス幅の Low 期間として CFLAG1 を 5 回カウントし、カウント終了時に CFLAG3 を出力する。CFLAG2 と CFLAG3 が JK-FF に入力され、デューティ=50%の PWM の波形が出力され、問題なく動作することを確認した。

4.2.3 PWM の FPGA 実装

ここで、これまでシミュレーションで確認してきたモデルの中から、代表的なマイコン周辺回路として 8 ビット PWM モデルを選択し、FPGA に実装して実験を行った。市販の ALTERA 社 StratixII 搭載 FPGA ボードを用い、別途作成した 8 ビット PWM の Verilog HDL モデルをコンパイルし、FPGA にマッピングした。図 4-41 にその回路図を示す。

また、図 4-42 に 8 ビット PWM に実装した動作を示す。今回の実験では、パルス幅のデューティ比を 0%から 100%の間でアップ/ダウン変化させ、これを繰り返し自走させる。また、実験用回路には FPGA ボードにあるスイッチを利用して、8 ビット PWM に任意に外部イベントを加えて、停止/再開させる HOLED_RELEASE/HOLED_REQUEST 信号の動作確認も行った。図 4-43 に FPGA に実装した 8 ビット PWM の観測した波形を示す。

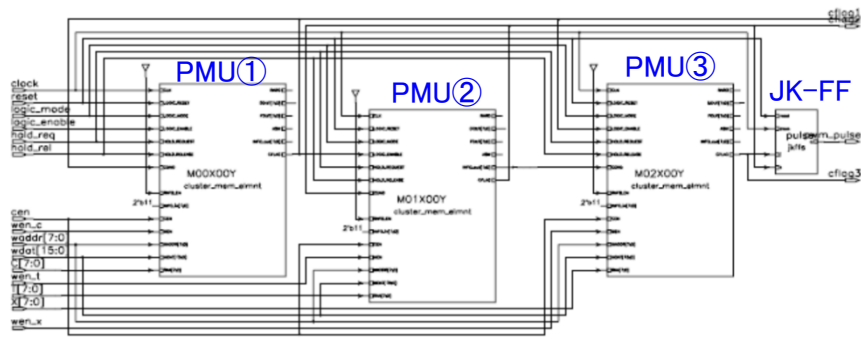


図 4-41 FPGA に実装した PMU を用いた 8 ビット PWM の回路図

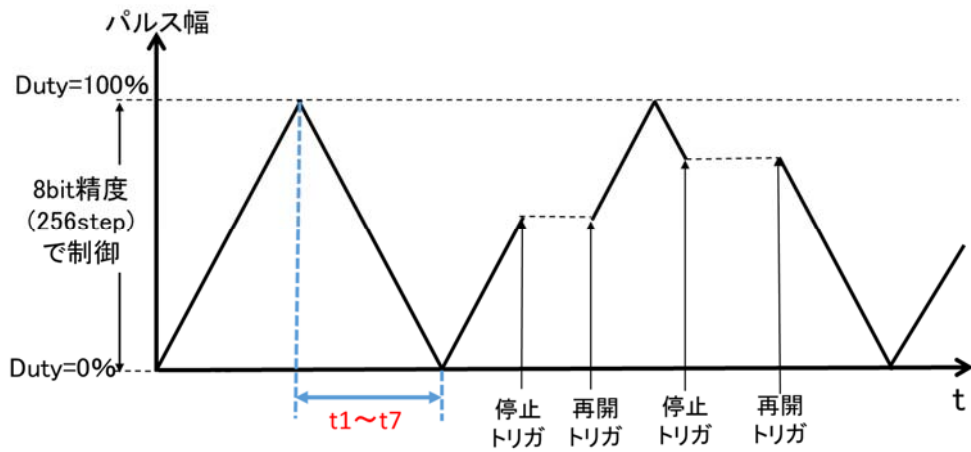
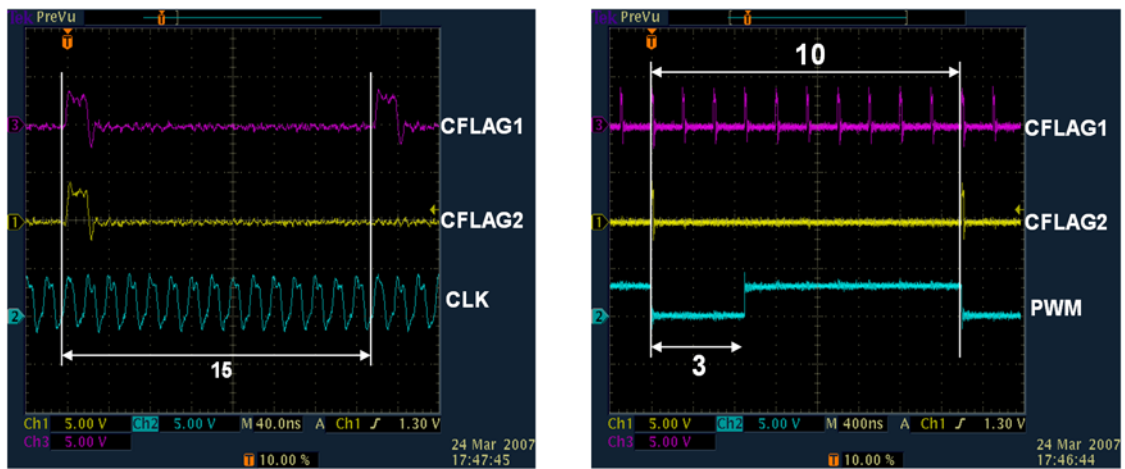


図 4-42 8 ビット PWM に実装した動作



(a)

(b)

図 4-43 FPGA に実装した 8 ビット PWM の波形

ここでは、Cレジスタ = 15 , Tレジスタ = 10 , Xレジスタ = 3 が設定された波形になっている。PMU①はシステムクロックを15分周した分解能の信号CFLAG1を出力し、PMU②はこのCFLAG1を10回カウント後、周期CFLAG2を出力する。PMU③は指定されたパルス幅のLow期間としてCFLAG1を3回カウントし、この結果、JK-FFからはデューティ70%のPWMの波形が出力されているのが観測できた。以上のようにシミュレーションモデルの波形と同様の動作が確認できた。

次に、FPGAへの実装結果を表4-12に示す。FPGA実装では、デバイスはALTERA社製EP1S40F780C58、ロジックシンセシスは同社から提供されるQuartusII Ver6.1を使用した。図4-44にこの実験で使用したFPGAボード写真を示す。動作時のシステムクロックは50MHzとした。また、FPGAボードにはALTERA社NiosIIのCPUが搭載されており、これを使用してPMUへのコンテキストの書き込みを行った。8ビット精度のPWMのFPGA実装では、4Kビット×3個、計12Kビットメモリと345個のLE(Logic Elements)を使って実装されている。ここで使用された8ビットのカウンタ/タイマ、すなわち1個のPMUは4Kビットのメモリと115個のLEで実装されている。

表 4-12 FPGA 実装結果

FPGA		ALTERA EP1S40F780C5
Logic Synthesis		ALTERA Quartus II 6.1
System Clock		50 MHz
PWM Func.	8 bit Accuracy	300 ns @ C=15
	256 Step/Cycle	3 μs @ T=10
	Number of LEs	345
	Memory	4K bits x 3

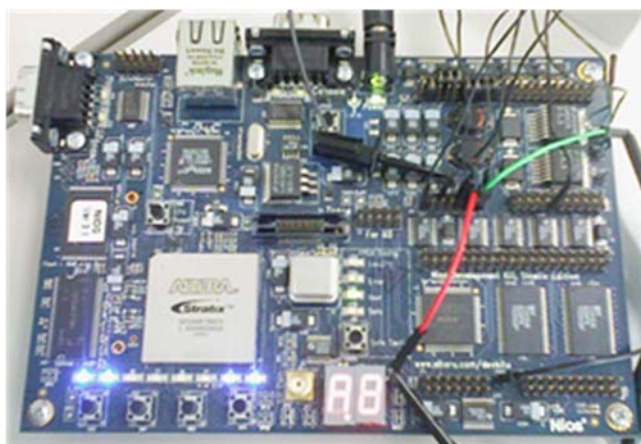


図 4-44 FPGA 実装ボード写真

以上、FPSMのシミュレーションモデルに実装し、動作検証したマイコン周辺回路機能を表4-13にまとめる。

表 4-13 マイコン周辺回路機能の検証結果

Peripheral		Option	Number of PMUs		
			× 2	× 3	× 8
Clocked Serial Interface	Receiver (Rx)	(8 bits)		✓	
	Transmitter (Tx)		✓		
PWM	8 bit Accuracy	1 output		✓	
	16 bit Accuracy	2 output			✓
FIFO Memory	R/W Buffer Size	up to 16 Bytes	✓		
		up to 256 Bytes		✓	

4.3 結言

FPSM アーキテクチャを提案し、SystemC によるシミュレーション検証を行った。基本論理素子 PMU のアーキテクチャのモデルベース設計を行い、PMU に必要なマイクロ命令/アドレス制御、入出力信号と I/O の定義、条件設定等のシミュレーション実験を繰り返すことでアーキテクチャの改良を行った。また、内蔵メモリとしての動作確認を行うとともに、マイコンで利用されるカウンタ/タイマ系、シフトレジスタ系および演算系の回路を PMU シミュレーションモデル上に実装して動作検証を行い、従来の回路と等価な動作を確認した。

次に、マイコン周辺回路機能のモデルとして、FIFO、シリアル通信インタフェース、PWM のシミュレーションモデルを構成し、動作確認を行った。確認した各マイコン周辺回路機と PMU の使用数を表 4-13 にまとめた。また、PMU シミュレーションモデルの SB 結線において、256 バイトバッファ FIFO では、出力されるフラグ出力数が増えたため、PMU の配置を工夫する必要が出てきた。この場合、使用する PMU は 3 個であるが、2 行にわたって配置する必要が出てきた。これは 1 行毎の出力段に JK-FF を 1 個のみ配置しているためであり、FULL および EMPTY フラグ 2 系統を JK-FF を経由して出力するため 2 行使った配置となった。このような実装ケースでは、実装効率の悪化を招くことが判明し、今後製品化する場合は、FPSM の出力段を工夫する必要がある。

さらにこの中から 8 ビット PWM を RTL 設計し、FPGA 上で実装評価を行った。市販の FPGA ボードを使用し、PMU 3 個で 8 ビット PWM を実装し、波形観測を行った。今回は SB 無しの実装であったが 8 ビット PWM シミュレーションモデルの波形評価結果と、RTL 設計した FPGA 実装した 8 ビット PWM の波形評価結果が、ともに設計通りの結果であった。

この結果から、FPSM アーキテクチャが内蔵メモリとして、かつマイコン周辺回路を再構成可能なプログラマブルロジックデバイスとして利用が可能であることを確認した。また、LSI 上に実装できる見通しを得た。

参考文献

- [4-1] 井口幸洋, 笹尾勤, 松浦宗寛, “算術分解を用いた基数変換回路の構成法(2),” 信学技報 RECONF2006-47, pp.19-24, 2006年11月.
- [4-2] 井口幸洋, 笹尾勤, 松浦宗寛, “算術分解を用いた基数変換回路の構成法(3),” 信学技報 VLD2006-135, pp. 97-102, 2007年3月.

第5章 実験チップの試作と評価

5.1 緒言

前章では FPSM アーキテクチャのシミュレーションモデル設計とその評価, およびマイコン周辺回路の FPGA 実装と動作検証について述べた. 本章では, この FPSM アーキテクチャの実験チップの設計および試作結果について述べる. 今回, $0.18\mu\text{m}$ CMOS 標準セルライブラリを用い, PMU 4×4 アレイ構成の FPSM 実験チップを論理設計・実装設計および実験チップ試作評価を行った. また, ターゲット回路として 16 ビットカウンタと FIFO を FPGA と FPSM 実験チップに実装した場合の実装面積, 消費電力の比較を行った結果について論じ, FPSM アーキテクチャの有効性を検証する.

5.2 FPSM の論理合成

前章で述べた FPSM アーキテクチャの仕様をもとに FPSM 実験チップのハードウェア設計を行った.

$0.18\mu\text{m}$, 1 層ポリシリコン, 5 層メタルの CMOS プロセスを用い, PMU 内の SRAM はライブラリのメモリモジュールを使い, 論理部分は論理合成で設計した. Verilog HDL で記述し, 論理合成には Synopsys 社の Design Compiler を用いた.

論理合成後の回路規模と面積を表 5-1 に示す. メモリ部は, 1 個の PMU 内のメモリが $256\text{ワード}\times 16\text{ビット}=4\text{K}\text{ビット}$ 構成の SRAM であり, 全体のメモリ容量は $4\text{K}\times 16=64\text{K}\text{ビット}$ となる. ロジック部の回路規模は 2 入力 NAND 換算で, 約 46k ゲートとなった. コア全体の面積は 1.86mm^2 で, このうちメモリが 1.28mm^2 , ロジック部分が 0.58mm^2 であった. ロジック部分は全体の 31%を占めている.

表 5-1 実験チップの回路規模と面積

Block		Gate Counts (gates)	Area (μm^2)
Per PMU	SRAM	256 w x 16 b	79800
	Logic	1352	17171
Per SB (Switch Box)		980	12451
MCU Interface		4792	60861
Total	SRAM	256 w x 16 b x 16 (64Kb)	127680
	Logic	46024 (46KG)	584629
Total			1861427

5.3 実装設計

今回, VDEC のチップ試作サービスを用いた. 利用可能なチップサイズは 2.5mm 角であり, この有効面積と配置配線マージンを考慮し, PMU 数は 4 行 \times 4 列の 16 個のアレイ配列とした. また, レイアウト設

計では Synopsys 社の自動配置配線ツール Astro を使用した。

図 3-24 のような、行配列で PMU を 1 行に 4 個配置する場合、MCU インタフェースとつながるバスや SB 間をつなぐグローバル配線が長くなり、動作周波数の低下や消費電力の増加につながる。このため PMU をマトリクス状配置することとし、そのフロアプランを図 5-1 に示す。PMU は SRAM 部と Logic 部を 1 ユニットとし、SB を挟んで対に配置されている。この対を 2 つ並べてマトリクス状に 1 行 4 個の PMU を配置している。更にこれを 2×2 のマトリクス状に配置し、中央部に MCU インタフェース部を配置している。この配置によりどの PMU もインタフェースからのバス配線がほぼ等距離になる。また、再構成の基本単位となる 4 個の PMU が最近接で配置されるため、SB 間のグローバル配線も短くなり、動作周波数の向上につながっている。

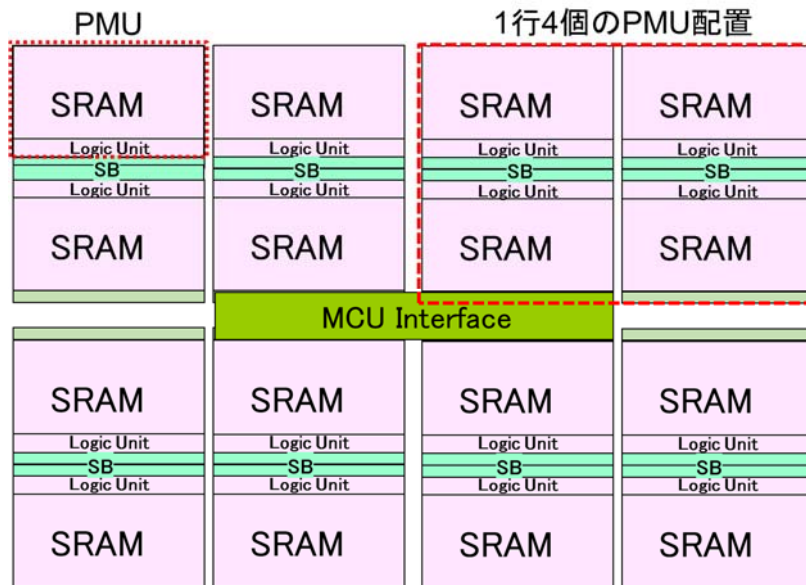


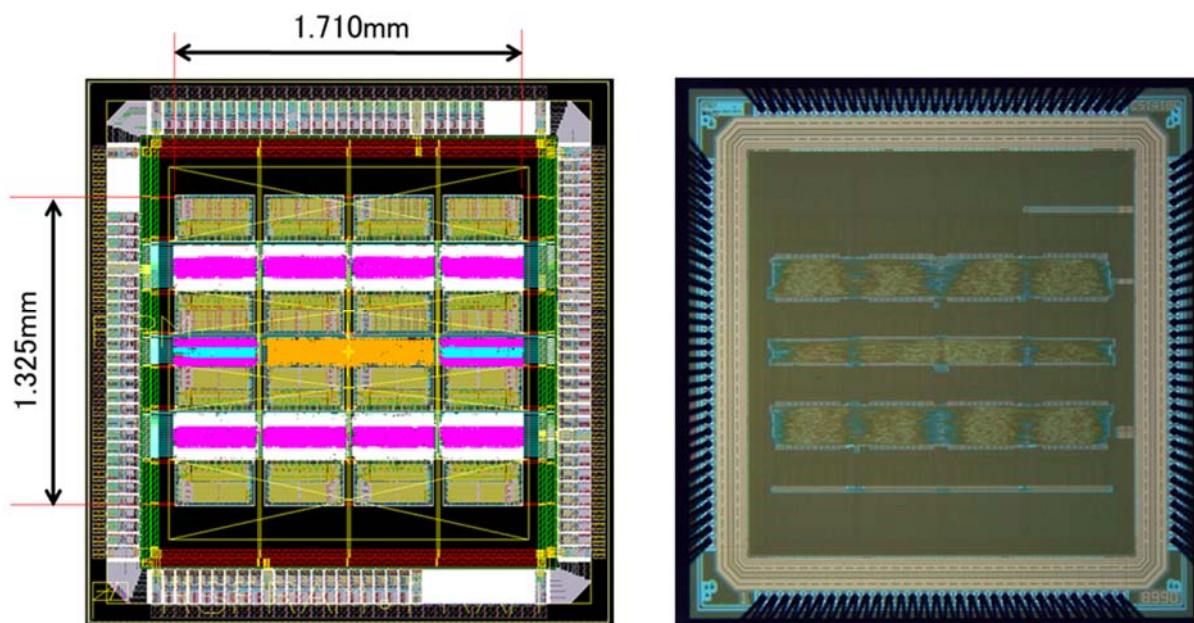
図 5-1 16 個の PMU からなる FPSM のフロアプラン

表 5-2 PMU アレイ配置の違いによる性能比較

アレイ構成	動作周波数 (MHz)		サイズ (mm ²)
	メモリ	周辺回路	
マトリクス状	69.9	61.1	2.37
行配列状	65.0	58.5	2.59

表 5-2 に、PMU アレイ配置による性能比較を示す。今回導入したマトリクス状のアレイ構成では面積が 9%ほど小さくなり、かつ動作周波数もメモリ動作では 7.5%、周辺回路動作では 5.1%ほど行配列状のアレイ構成よりも高速であることがわかった。

以上によりマトリクス状にアレイ配置した FPSM 実験チップのレイアウトプロットを図 5-2 (a) に、実験チップ写真を図 5-2 (b) に示す。



(a) レイアウトプロット図

(b) 実験チップ写真

図 5-2 レイアウトプロット図と実験チップ写真

5.4 試作と評価

今回試作, 評価した実験チップの諸元を表 5-3 に示す. $0.18\mu\text{m}$, 1層ポリシリコン, 5層メタルの CMOS プロセスを用い, PMU 4×4 アレイ構成の FPSM の設計を行った. PMU の SRAM はライブラリとして準備されているメモリモジュールを利用した. 実験チップのゲート規模は 2 入力 NAND 換算で 46k ゲート, コア部の面積は 2.265mm^2 である (ここで表 5-1 の面積 1.86mm^2 との差 0.41mm^2 は配線や空き領域によるものである). 実験チップの消費電力は, 目標動作周波数 50MHz, 電源電圧 1.8V において, 基本論理素子 PMU 単位で約 1mW が得られた. また, shmoo plot により, 電源電圧 1.8V 時で最大動作周波数 61.5MHz 動作を確認した.

表 5-3 実験チップの諸元

プロセス技術	0.18 μm , 1層ポリシリコン, 5層メタル CMOS
電源電圧	コア部: 1.8V, I/O部: 3.3V
コア部のチップ面積	2.265mm^2 (1.325mm x 1.710mm)
PMUのチップ面積	0.0988mm^2 (0.380mm x 0.260mm)
SRAM 構成	256 word x 16 bits x 16
最大動作周波数	61.5MHz @ 1.8V
消費電力/PMU	1.1 m Watt @ 50MHz

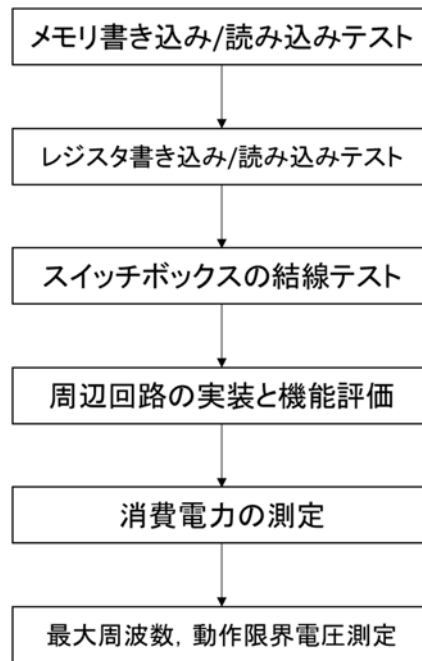


図 5-4 実験チップの評価フロー

図 5-4 に実験チップの評価フローを示す。まず、評価環境を整え、チップ内のメモリ、レジスタの書き込み/読み出しテストおよび SB の結線テストを行い、動作確認を行った。また、表 5-4 に示す論理演算回路、マイコン周辺回路のコンテキスト、SB の結線情報を書き込んで動作確認を行った。次に動作限界周波数及び動作限界電圧を確認するために 5 個のサンプルで shmoo plot を取り、各回路の消費電力を測定した。

5.4.1 周辺回路機能の実装評価

以下、周辺回路の実装・機能評価、消費電力および最大周波数、動作限界電圧の測定について述べる。第 4 章でシミュレーションした基本論理演算回路、マイコン周辺回路を今回試作した実験チップ上に実装し、機能評価を行った。今回検証した各周辺回路機能の仕様とその機能に使用される PMU の数を表 5-4 に示す。カウンタ/タイマおよびシフタの仕様は 8 ビット毎にビット長を可変し、それに伴い PMU の数が増えている。ここでは、8 ビットカウンタは PMU1 個、32 ビットカウンタは 4 個の PMU を使って実装した。表中の“✓”は動作確認済みであり、当初想定した周辺回路機能が全て正しく動作する事を確認した。また、この中から代表的な周辺回路機能として、8 ビットカウンタ、16 ワード FIFO、シリアル送信インタフェース、16 ビット精度 PWM を実験チップに実装して波形観測した結果を図 5-5～図 5-8 に示す。

図 5-5 の 8 ビットカウンタでは、16 カウント毎にカウント完了フラグを繰り返し発行されているのが観測できた。次に、図 5-6 では 16 ワード FIFO の基本動作として、16 回書き込み/読み込みを繰り返し実行させ、16 回書き込みで Full フラグ、16 回読み込んで Empty フラグを交互に発行していることが確認できた。図 5-7 のシリアル通信の送信側では、送信信号“01010101”を逐次 1 ビットずつ送信し、送信完了後に、送信完了フラグが出力されているのを観測した。

最後に、図 5-8 では 16 ビットカウンタを用いた 16 ビット PWM2 相出力の波形を示す。それぞれ設定したデューティ比 40%および 20%の PWM 波形が観測された。

以上の如く、表 5-4 に示した周辺回路の動作確認を行い、問題なく動作する事を確認した。

表 5-4 実験チップ上で実装評価した周辺回路機能

周辺回路機能	仕様	PMU使用数				
		X1	X2	X3	X4	X8
Counter / Timer (8/16/24/32bit構成)	Up counter	✓	✓	✓	✓	—
	Down counter	✓	✓	✓	✓	—
Shifter (8/16/24/32bit構成)	Logic Shift	✓	✓	✓	✓	—
	Arithmetic Shift	✓	✓	✓	✓	—
Clocked serial Interface	Receiver	—	—	✓	—	—
	Transmitter	—	—	—	✓	—
FIFO memory	16/256 words	—	✓	✓	—	—
Calculation unit	Adder (4/8/16 bits)	✓	✓	✓	✓	—
	Subtractor (4/8/16 bits)	✓	✓	✓	✓	—
PWM	8-bit accuracy	—	—	✓	—	—
	16-bit accuracy	—	—	—	—	✓

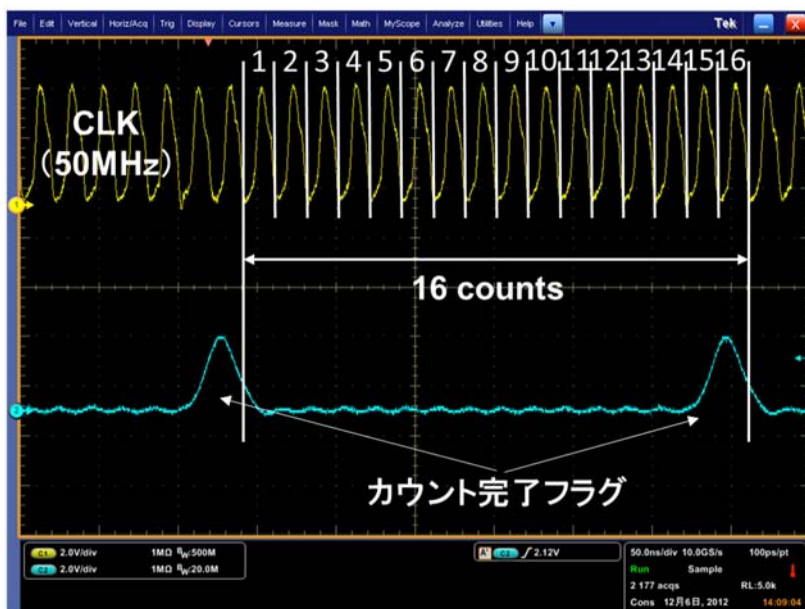


図 5-5 8 ビットカウンタ構成時の波形 (16 カウント繰り返し)

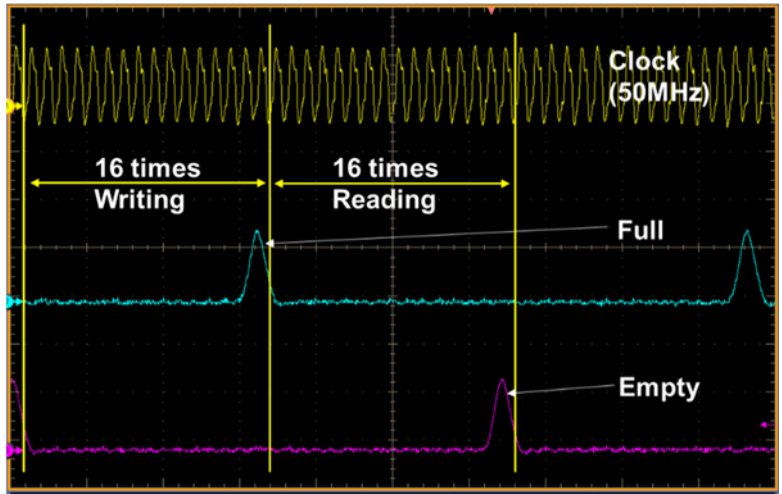


図 5-6 16 ワードバッファ FIFO 構成時の波形 (16 回書き込み/読み込み繰り返し)

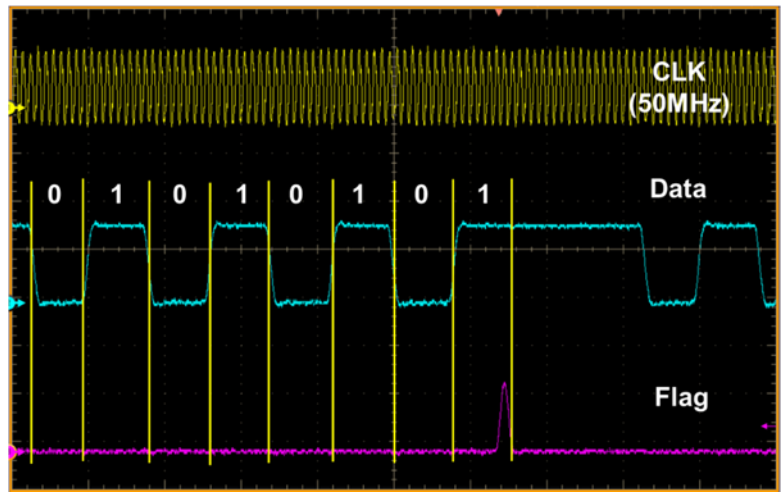


図 5-7 シリアル送信側構成時の波形 (“01010101” を順次送信する)

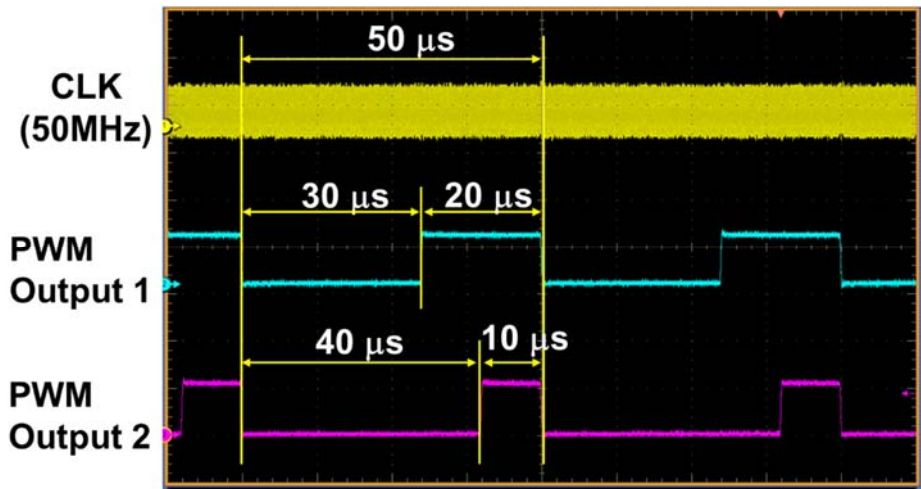


図 5-8 16 ビット精度 PWM 構成時の波形 (Duty 比 40%, 20%)

5.4.2 消費電力測定

5個のサンプルを使って、消費電力の測定を行った。各周辺回路の動作時に流れる電流値を実験チップ上で測定し、この平均値に電源電圧 1.8V を積算し、消費電力を算出した（ただし、I/O 部の消費電力は含まない）。各周辺回路で使用される PMU の数と算出した消費電力の平均値を表 5-5 に示す。測定時のクロック周波数は 50MHz である。

この結果から、1 個の PMU の消費電力は約 1mW であり、使用する PMU の数に比例して消費電力が増加していることがわかる。また、これらのバラツキはサンプルのバラツキだけでなく、PMU の利用信号数、SB の配線数も関連すると思われる、今後の詳細な分析が必要と考える。

表 5-5 各周辺回路の消費電力と PMU の数（クロック周波数：50MHz）

周辺回路	PMU 数と消費電力 (mW)				
	1	2	3	4	8
カウンタ	1.202	2.186	3.131	4.145	-
シフタ	1.012	2.025	3.042	4.092	-
演算器	1.012	2.050	3.111	4.158	-
シリアル	-	-	3.208	4.183	-
FIFO	-	2.528	3.471	-	-
PWM	-	-	3.290	-	8.500
平均	1.075	2.197	3.209	4.145	8.500

5.4.3 shmoo plot 評価

サンプル 5 個の内 1 個の実験チップの shmoo plot を測定した結果を図 5-9 に示す。

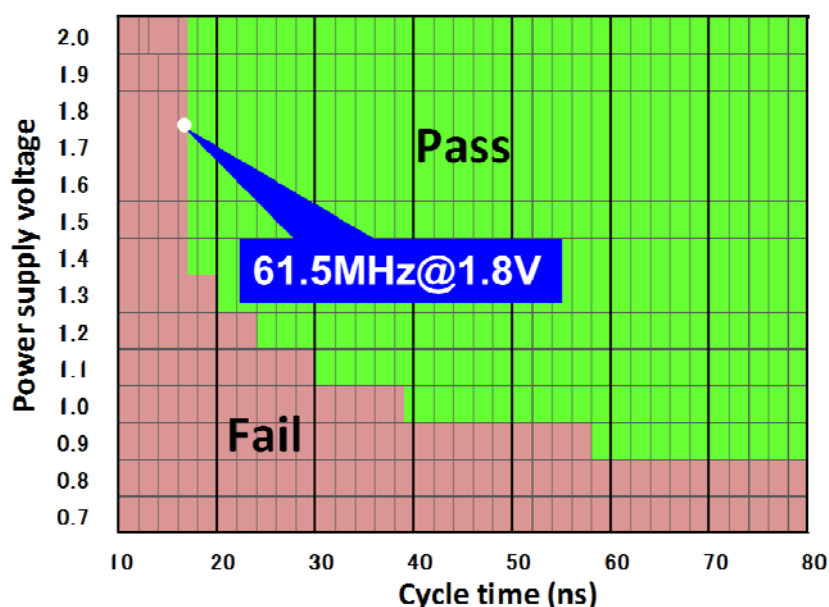


図 5-9 16 ビット精度 PWM を実装して測定した shmoo plot

最も動作負荷が掛かるマイコン周辺回路として PMU を 8 個使用する 16 ビット精度 PWM を実験チップ上に実装し、波形が正しく観測できた場合を “pass” として計測を行った。この結果、電源電圧 1.8V 時の最大動作周波数は 61.5MHz であった。また、動作限界電圧は約 0.9V であることを確認した。

5.5 FPGA との比較

FPSM で実現した周辺回路の面積と消費電力を比較評価するため、実験チップから得られた PMU の評価結果を用い、同じ周辺回路機能を実装する FPGA と実装比較の考察を行った。

表 5-3 より、PMU 部のチップサイズは $0.380\text{mm} \times 0.260\text{mm} = 0.0988\text{mm}^2$ である。このうちアドレス制御部は 0.017mm^2 、SB は 0.0124mm^2 および MCU インタフェースは 0.06mm^2 である [5-1]。ここで MCU インタフェースは共通部であり、FPSM に 1 個しかないので、無視できるものとした。

表 5-6 に 16 ビットカウンタ (フリーランタイマ) と 256 ワード \times 8 ビット FIFO を選択し、実装比較を行った結果を示す。FPSM では、どちらも 2 個の PMU で実装可能である (FIFO のデータメモリは含まない)。実装面積は、上述で示した PMU と SB を加算したものの 2 倍となる。PMU 1 個の実装面積が $0.0988\text{mm}^2 + 0.0124\text{mm}^2 = 0.1112\text{mm}^2$ であり、ここでは 2 個を使用するため、実装面積は、2 倍の 0.222mm^2 となる。

また、表 5-5 より $1\text{mW}/\text{PMU}$ (@50MHz) であり、動作周波数 20MHz においては、 $0.4\text{mW}/\text{PMU}$ (@20MHz) とした。ここで、PMU の特徴でもあるが、16 ビットのカウンタを実装した場合、下位 8 ビットの活性化率は 100% であるが、上位 8 ビットのカウンタは $1/256$ の活性化率しかないので、消費電力はほぼ 1 個の PMU の消費電力とみなしてよい。これは FIFO も同様である。したがって、どちらも消費電力は 0.4mW となる。

表 5-6 FPSM と FPGA の実装比較

Function	FPSM			FPGA		
	No. of PMUs	Area (mm ²)	Power ¹⁾ (mW@20MHz)	No. of ²⁾ ALMs	Area ³⁾ (mm ²)	Power ³⁾ (mW@20MHz)
Free Run Timer (16 b)	2	0.222	0.4	9	0.23	1.12
FIFO ⁴⁾ (256W \times 8 b)	2	0.222	0.4	19	0.50	2.36

- 1) 電力は、ほとんどのサイクルにわたって 1 個の PMU しか動作しないので、PMU の数とは無関係
- 2) ALTERA 社 Stratix II の実装実験結果
- 3) Ref. 2 の $0.22\mu\text{m}$ データから変換したデータ [5-2]
- 4) データメモリは含まない

Stratix と Cyclone の LE の基本構造がほぼ同じであり、今回は ALTERA 社 Stratix II シリーズ [5-4] の FPGA を比較対象とし、面積と消費電力の評価を行った。ここで、Stratix II の基本モジュールである Adaptive Logic Module (ALM) の上述の回路実装に必要な ALM 数を ALTERA 社のコンパイラツール Quartus II 32-bit Version 13.0.1 を使って取得した。Stratix II に 16 ビットカウンタ (フリーランタイマ)

と 256 ワード×8 ビット FIFO を実装するため、コンパイラツールを使ってコンパイルした結果、16 ビットカウンタ（フリーランタイマ）実装では、ALUT が 9 個、256 ワード×8 ビット FIFO 実装では 19 個使用する結果が出た。ここで、Xilinx 社 Vertex のスライスは Stratix II ALM にほぼ等しいため [5-3] [5-4]、 $CLB = 2 \text{ スライス} = 2ALM$ と仮定し、Xilinx 社 CLB の数に変換する。以上の仮定のもと、ALM の面積と消費電力を Xilinx 社 Vertex のデータから変換し、導き出すこととした（参考文献 [5-2] の表 II を参照）。

0.22 μm のプロセス技術で、動作周波数 20MHz、電源電圧 2.5V で、CLB あたりの平均面積と平均消費電力はそれぞれ 0.0780mm² と 0.587mW（標準偏差は 0.022mW）である。ここで、ALM の密度が上記の CLB の半分であると仮定し、0.18 μm プロセス技術における ALM の面積は、Ref [5-2] の CLB の平均面積に $(0.18 \mu\text{m} / 0.22 \mu\text{m})^2$ を掛けて算出することとした。

したがって、9 個の ALM を使用する 16 ビットカウンタの面積は、0.23mm²、19 個の ALM を使用する 256 ワード×8 ビット FIFO の面積は、0.50 mm² となる。消費電力は fcv2 を考慮すると、 $(0.18 \mu\text{m} / 0.22 \mu\text{m})$ と $(1.8 \text{ V} / 2.5 \text{ V})^2$ のデータを掛け合わせるにより、0.22 μm のデータを 0.18 μm のデータに変換し算出し、それぞれ 1.12mW、2.36mW となる。

以上により、16 ビットカウンタでは FPSM は FPGA とほぼ同等の実装面積で消費電力は約 1/3、256 ワード×8 ビット FIFO では FPGA の実装面積の約 1/2 以下で消費電力は約 1/5 以下となった。また、FPGA 上では、16 ビットカウンタと 256 ワード×8 ビット FIFO の実装を比較すると、FIFO では、面積が約 2 倍、消費電力も約 2 倍となっているが、FPSM ではどちらも同じ面積と消費電力で実現できている。

5.6 結言

0.18 μm の CMOS プロセスを用いて、FPSM の実験チップを試作した。実験チップのゲート規模は 2 入力 NAND 換算で 46k ゲート、コア部の面積は 2.265mm² である。カウンタ/タイマ、シフタ、シリアル通信インターフェース、FIFO、PWM 等、想定した周辺回路機能が全て再構成でき、かつ動作することを確認した。

実験チップの消費電力は、目標動作周波数である 50MHz、電源電圧 1.8V において、基本論理素子 PMU 単位で約 1mW が得られた。また、shmoo plot により、電源電圧 1.8V 時で最大動作周波数 61.5MHz 動作を確認するとともに、動作限界電圧が 0.9V であることを確認した。

さらに、今回限定的ではあるが、16 ビットカウンタと 256 ワード×8 ビット FIFO を FPSM および FPGA において実装比較を行い、カウンタおよび FIFO に関しては、FPSM は FPGA の実装面積の同等か半分以下、さらに消費電力については FPGA の約 1/3 から 1/5 程度少ない消費電力になることを確認した。限定的な実装回路機能となるが、実装面積および消費電力ともに FPGA よりも FPSM アーキテクチャの方が優位であり、マイコン向けプログラマブルロジックデバイスとして十分利用可能と考える。

参考文献

- [5-1] Y. Kawamura, N. Okada, Y. Matsuda, T. Matsumura, H. Makino, and K. Arimoto, “A Field Programmable Sequencer and Memory with Middle Grained Programmability Optimized for MCU Peripherals,” IEICE Trans. Fundamentals, Vol. E99-A, No. 5, pp. 917-928, May 2016.
- [5-2] K. Nakamura, T. Sasao, M. Matsuura, K. Tanaka, K. Yoshizumi, H. Nakahara, and Y. Iguchi, “A memory-

based programmable logic device using look-up table cascade with synchronous static random access memories,”
Japanese Journal of Applied Physics, vol.45, no.4B, pp.3295-3300, Apr. 2006.

[5-3] https://www.altera.com/content/dam/alterawww/global/en_US/pdfs/literature/hb/stx2/stratix2_handbook.pdf,
“Stratix II Device Handbook, Volume 1,” p. 2-7.

[5-4] http://www.xilinx.com/support/documentation/data_sheets/ds003.pdf,
“Virtex 2.5V Field Programmable Gate Arrays, Product Specification,” p. 5.

第6章 パケットフィルタ応用

本章では、PMU アーキテクチャをベースにしたハッシュ探索を行う回路を提案し、一致/不一致検出回路の一致検出回路と組み合わせることで、高スループット、かつ低消費電力なパケットフィルタ回路を開発した。ここでは PMU の応用部分を述べ、さらに提案したパケットフィルタを実装した TEG チップの試作・評価した結果について述べる。

6.1 緒言

近年、インターネットの普及とともにネットワークの通信量も急激に増加しており、これに伴い、住宅用のルーターやゲートウェイ等の通信端末も高スループット、低消費電力化が求められている。

この低消費電力化の鍵となるのが、パケットフィルタの検索エンジンである。この検索エンジンは、入力パケットが予め登録されているルールと一致するか否かを判定する。そのためには登録されているルールをメモリから読み出す必要がある。従来の線形探索法では、順次ルールと比較するため、最終的な一致/不一致の結果を得るには、頻繁にメモリにアクセスすることになる。これによりスループットの低下や消費電力の増加を招く。また、このメモリアクセス数を減らすために、様々なアルゴリズムが提案されてきた [6-1]-[6-6]。この改善策として従来パケットフィルタ回路に不一致検出回路を加え、スループットを向上させる技術が提案されている [6-7][6-8][6-9]。

6.2 パケット検索方式

一般のパケット検索では、IP アドレス等の照合データと事前に登録した一致条件とを照合して一致/不一致の判定を行い、これに基づき、パケットの破棄、透過等の処理を行う。この一致検出部には線形探索、二分探索、ハッシュ関数利用など様々な方式が用いられている。線形探索方式の代表的な半導体デバイスは Content Addressable Memory (CAM) である。また、二分探索は専用ハードウェアまたはソフトウェアが用いられ、ハッシュ関数を利用する場合は、主にソフトウェアで実装される。これらは、それぞれ表 6-1 のような特徴を持ち、利用される機器によって最適な方式が採用される。

表 6-1 各検索方式の特徴

検索方式	メモリアクセス
線形探索	一致: ルールの数の半分 (Ave.) 不一致: すべてのルール数
二分探索	メモリへのアクセス数が半分 (線形探索との比較)
ハッシング	1つのバケット内でメモリアクセスを減らす がルール数は限られる (バケット容量はハッシュ衝突に起因)

6.3 一致/不一致検出パケット検索エンジン

不一致検出はパケットと一致条件との部分的な一致を見る事で、1 サイクルで不一致判定ができる仕組みである。一致/不一致検出回路を並列動作させることで、従来の線形探索方式のパケットフィルタ回路より高スループットが実現できる。今回、IPV6 を想定するとともに、スループット 40Gbps を目標 [6-10] に開発を行った。

一般に不一致判定は、全一致条件の照合が完了した後で、不一致が判定されるため、一致判定に比べ、時間がかかる。そこで、照合データを全一致条件との照合を完了する前に、不一致検出回路を用いて、不一致判定ができれば、判定時間を短縮でき実行スループットを向上することが可能となる。図 6-1 に今回提案する一致/不一致検出を用いたパケット検索エンジンの構成を示す。

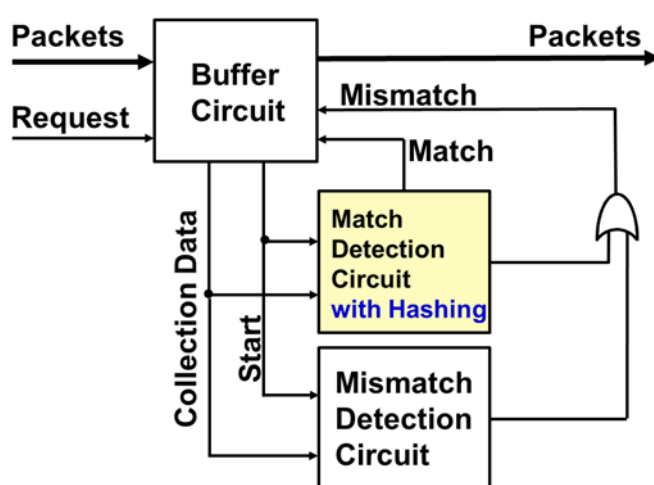


図 6-1 一致/不一致検出回路を用いたパケット検索エンジンの構成

検索エンジンはバッファ回路、一致検出回路および不一致検出回路で構成される。バッファ回路は、受信パケットからヘッダの全部または一部を抽出し、照合データとして一致検出回路と不一致検出回路へ Start 信号とともに送付する。一致検出回路と不一致検出回路が同時に照合データと登録ルールとの比較を開始し、不一致検出回路が照合データと登録ルールが不一致であることを検出した場合、不一致 (Mismatch) 信号が出力され、一致検出回路は検索動作を停止し、次の受信パケットの検索を開始する。不一致検出回路から一致信号 (Match) を受信した場合は、一致検出回路が一致/不一致判定が最終的に得られるまで照合データと登録ルールを比較し、一致と判定されたパケットは廃棄される。

6.3.1 不一致検出回路

図 6-2 に不一致検出回路を示す。不一致検出回路では通常、照合データとルールテーブルの長さは 2^{16} ワード×1 ビットのサイズが必要となる。IPV6 を想定した場合、 2^{512} ワード×1 ビットといった膨大なメモリサイズが必要となる。このメモリサイズを小さくするため、ルール長 512 ビットを 8 ビット毎に分割、64 個の複数のメモリに分割して照合データと比較することによりメモリサイズを 16K ビットに削減できる。このようにメモリを分割した場合、どのルールとも一致しないにもかかわらず、一致の判定を

行う場合がある。不一致検出回路で一致が検出された場合は、一致判定回路が全一致条件と照合を行い、最終的に一致/不一致を判定する。

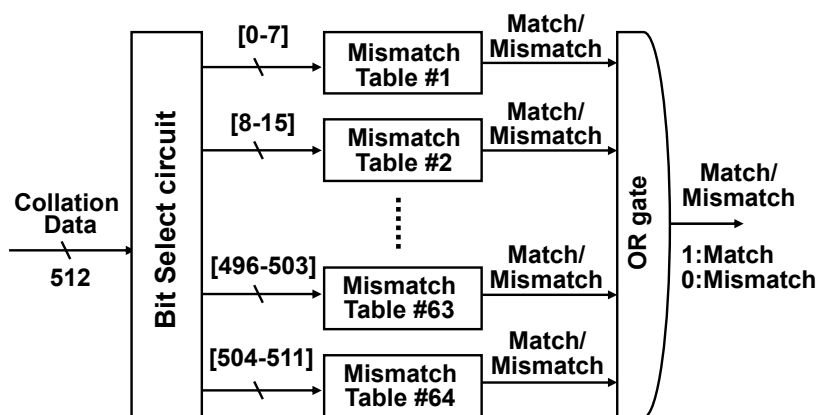


図 6-2 不一致検出回路

6.3.2 一致検出回路

図 6-3 に一致検出回路を示す。一致検出回路はインデックステーブル、ルールテーブル、比較器および制御回路で構成される。また、インデックステーブルとルールテーブルは RAM で構成され、ハッシュ探索機能を実現している。

インデックステーブルの定義を表 6-2 に示す。一致条件の先頭 8 ビットをアドレスとし、そのアドレスと同じ一致条件がインデックステーブル上に登録されているか否かの Match Flag (MF: 1 ビット情報) と、登録済みの場合には、その一致条件が格納されているルールテーブルの Index Address (IA: 9 ビットのアドレス) を格納する。インデックステーブルのメモリ容量は 256 ワード×10 ビット、すなわち約 2.5K ビットとした。

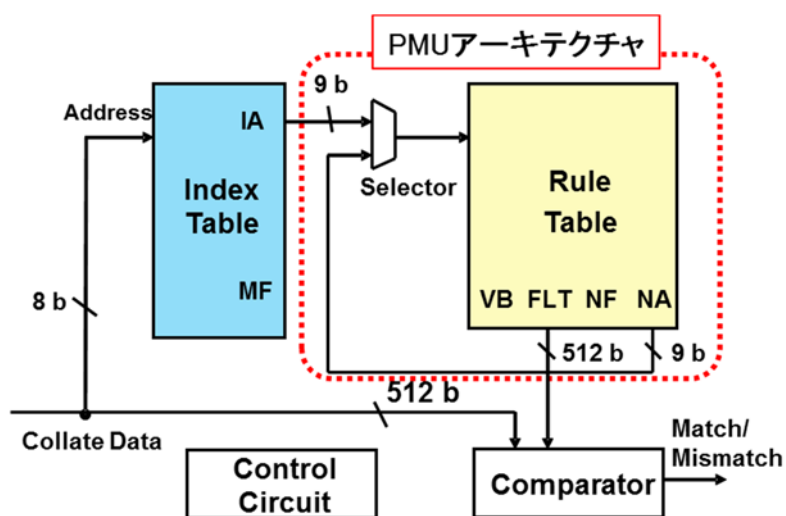


図 6-3 一致検出回路のインデックステーブルおよびルールテーブル

表 6-2 インデックステーブルの定義

MF(1bit)	IA(9bit)
1(登録)	Add of Rule Table
0(登録無)	0

表 6-3 ルールテーブルのフラグフィールドの定義

VB(1bit)	NF(1bit)	NA(9bit)
1(登録)	1	Next address in the same set
1(登録)	0	0: Last rule in the same set
0	1	Next empty address
0	0	0: Full

PMU アーキテクチャを導入したルールテーブル部は、入力セクタ部とメモリ部で構成される。このルールテーブルの RAM 部は PMU と同様、データフィールド（以下、FLT）には 512 ビットの一一致条件ルールデータ、フラグフィールドには 1 ビットの Valid bit (VB) 信号、Next Flag (NF) 信号および 9 ビットの Next Address (NA) がセットされる。登録される一一致条件ルールは、先頭の 8 ビットで分類され、先頭 8 ビットが共通である一一致条件ルールは 1 つのグループとみなす。VB は、FLT に一一致条件ルールが登録されているかどうかを示し、VB = 1 で登録有、VB = 0 で登録無を意味する。NF は、同じグループに属するルールが以降のアドレスに格納されているか否かを示し、格納されている場合は、NF = 1 となり、この場合は NA フィールドに次の条件が格納されているルールテーブルのアドレスを格納する。同じグループに属するルールが以降のアドレスに無い場合は、NF=0、NA=0 が格納される。表 6-3 にこれらフラグフィールドの制御信号の定義を示す。

ルールテーブルのメモリ容量は、最大 512 ルールの登録を前提に、1 ビットの VB、512 ビットの FLT、1 ビットの NF および 9 ビットの NA フィールドとし、512 ワード×523 ビット、すなわち、約 261K ビットとなる。

【PMU を利用したリンクリストハッシュテーブルの動作】

ここで、一致検出回路に適用する PMU の利用方法について述べる。図 6-4 に PMU アーキテクチャを利用し、ルールテーブル上に実装したリンクリストハッシュテーブルの動作を示す。図中の “/” はアドレスが存在しない状態を示す。上述のインデックステーブルに先頭 8 ビットに分類されたデータが入力し、インデックステーブルがアクセスされる。ルールが登録されている場合 (MF=1) は、PMU へのアクセスアドレスが存在し、PMU に実装されたルールテーブルがアクセスされる。例えば、ハッシュのグループ Z の場合、まず先頭の Z1 がアクセスされ、ルールデータをコンパレータに出力する。この時、NF=1 の場合、グループ Z のルールが複数あり、NA で示されたルールテーブル内のアドレスに Z2 の値が記憶されていることを示す。このアドレスは PMU 内の帰還ループと入力選択セクタを經由し（図 6-3）、PMU 内のこの NA のアドレスをアクセスし、Z2 のデータをコンパレータに出力する。上述と同様に NF=1 であれば、次

の Z3 のデータが存在し、再び NA に示されたアドレスをアクセスし、Z3 のデータをコンパレータに出力する。この時、NF=0 で NA が “/” になるとグループ Z の終端を意味し、ハッシュのグループ Z の探索は終了し、次のインデックステーブルからのアクセスを待ため、入力選択セクタが帰還ループからインデックステーブルの入力に切り替わる。X1 がアクセスされればグループ X を、Y1 がアクセスされればグループ Y の探索が行われる。

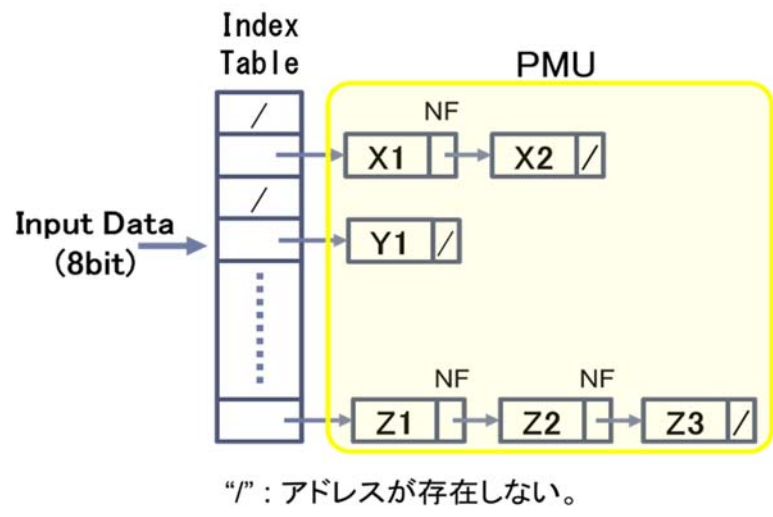


図 6-4 PMU を利用したリンクリストハッシュテーブルの動作

一致検出回路の具体的な動作を図 6-5 に示す。一致検出の動作は、基本的動作は上述の PMU を利用したリンクリストハッシュテーブルの動作と同じである。先頭 8 ビット「a」の条件が 2 つある場合、「Rule a0」, 「Rule a1」で表される。一致検出回路に入力された受信パケットは、バッファ回路を經由して、全部またはその一部が抽出され、照合データとして、一致判定回路および不一致判定回路に送られ、判定動作を同時に開始する。ここで IA, ルールテーブルおよび NA のアドレス値は説明上 10 進数で表記している。

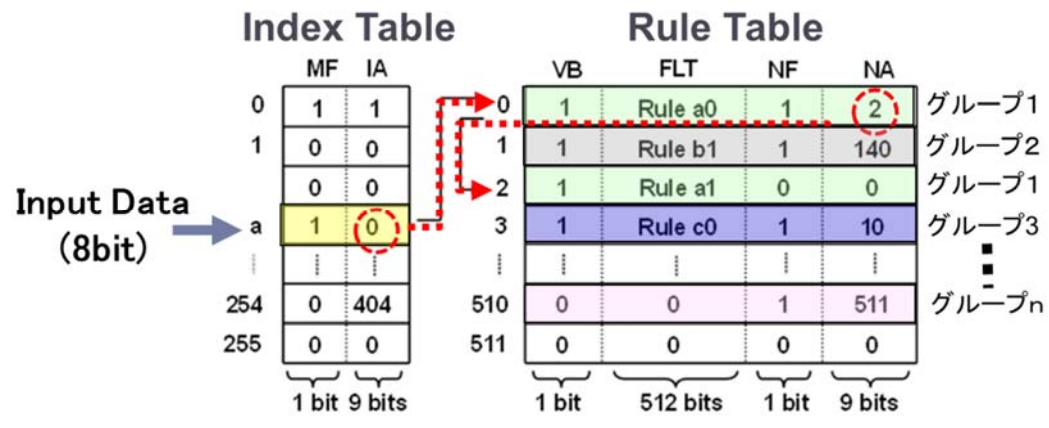


図 6-5 インデックステーブル・ルールテーブルの動作

一致検出回路のインデックステーブルは、バッファ回路から照合データの最初の8ビットでアクセスされ、同じ最初の8ビット列でルールがルールテーブルに登録されているか否かを判定する。同じ先頭8ビットがルール登録されていない場合 (MF=0) は、一致検出は完了する。ルールが登録されている場合 (MF=1), 最初のルールが IA フィールドのアドレスを参照し、ルールテーブルから読み出される。これらが合致している場合、ルールは比較回路と一致検出で照合データと比較され終了する。次に合致しない場合、同じグループに属する後続のルールが登録されていない場合 (MF=0) は一致検出動作が終了する。ルールが登録されている場合 (MF=1) は、次のルールが NA フィールドと照合データを参照して読み出され、ルールと比較される。この手順を NF=0 になるまで繰り返す。

具体的には、インデックステーブルに8ビットの入力データ“a”がアドレスとして入力される。このデータフィールドには、MF=1が登録されており、ルールテーブルのアドレスが存在する事を示し、そのIAのアドレス値“0”で、ルールテーブルをアクセスする。ルールテーブルのアドレス“0”にある512ビットのRule a0が読みだされコンパレータに出力する。この時、VB=1, NF=1が登録されており、表6-2に示すように同じハッシュのグループが存在していることを示し、NAに登録された次の9ビットアドレスが帰還ループ、入力選択セレクタを経由し、再びルールテーブルをアクセスする。この場合、NA=2であり、ルールテーブルのアドレス“2”をアクセスし、上記と同様に512ビットのRule a1をコンパレータに出力する。また、この時VB=1, NF=0が登録されており、表6-2に従いこのグループの最後であることを示し、ルールテーブルの帰還ループを使ったアクセスが停止し、入力選択セレクタは再びインデックステーブルの入力に切り替わり、次の入力を待ちとなり、次の探索に遷移する。

6.4 スループット評価

提案する検索エンジンの性能をシミュレーションで評価した結果について報告する。パケットフィルタの Throughput (TP) を1秒間に判定処理できるパケット数で定義する。1個のパケットを判定するために必要な平均サイクル数を Cycle per Packet (CP) とすると、 TP は、

$$TP = f \text{ (クロック周波数)} / CP, \quad (1)$$

で表される。登録されるルールおよび入力される照合データはランダムで、照合データとルールは n ビットで、 m ビットずつ、 $L (=n/m)$ 個に分割する。従って不一致メモリの数は L 個である。登録されているルールの数は N 個とすと、照合データの m ビットが、1 個の一致条件の対応する m ビットと一致する確率は $(1/2)^m$ なので、 i 番目の不一致テーブルにおいて N 個のルールのどれとも一致しない確率 p_i は、

$$p_i = (1 - (1/2)^m)^N. \quad (2)$$

となる。したがって、 L 個の不一致テーブルで、一致と判定される確率は $(1-p_i)^L$ となり、不一致検出率 P は、

$$\begin{aligned} P &= 1 - (1 - p_i)^L \\ &= 1 - \{1 - (1 - (1/2)^m)^N\}^L. \end{aligned} \quad (3)$$

で表される。提案するアーキテクチャでは、先頭8ビットで条件を分類しているので、一つのグループに属する一致条件の数は平均して $N/256$ 個である。その時 CP は、

$$CP = 1 + (1 - P) \frac{N}{256}, \quad (4)$$

となり，TP は式(5) で与えられる．

$$TP = \frac{100}{1 + (1 - P)N/256} \text{ [packet/s]}, \quad (5)$$

ただし，不一致メモリ，索引テーブル，ルールテーブルは 100MHz で動作するとした．不一致判定回路で一致と判定され，一致判定回路に最終判定をゆだねられたパケットが，条件テーブルで一致と判定される確率は $(1/2)^{512}$ と非常に小さい．したがって同じグループに属する条件，つまり平均して $N/256$ 個の一致条件と全て照合されると単純化している．不一致判定回路が無い場合は，式(5)で， $P=0$ であり，

$$TP = \frac{100}{1 + N/256} \text{ [packet/s]}, \quad (6)$$

となる．

Verilog 記述による RTL モデルを用い，TP をシミュレーションで評価した．照合データとルールは全て 512 ビット (=n) で，8 ビット (=m) ずつ 64 (=L) 分割している．登録条件数は 512 個 (=N) ある．512 個の条件を乱数で生成し，そのうち $64 \times k$ ($k=0, 1, \dots, 8$) 個の条件を登録し，別途乱数で発生させた 5,000 個の照合データを用いて判定処理を行った．図 6-5 に TP のシミュレーション結果を示す．不一致検出回路を組み合わせた回路では，式(3)より明らかなように $P \approx 1$ であるから，式(5)から $TP=100M$ [packet/sec] となり，シミュレーション結果と良く一致している．

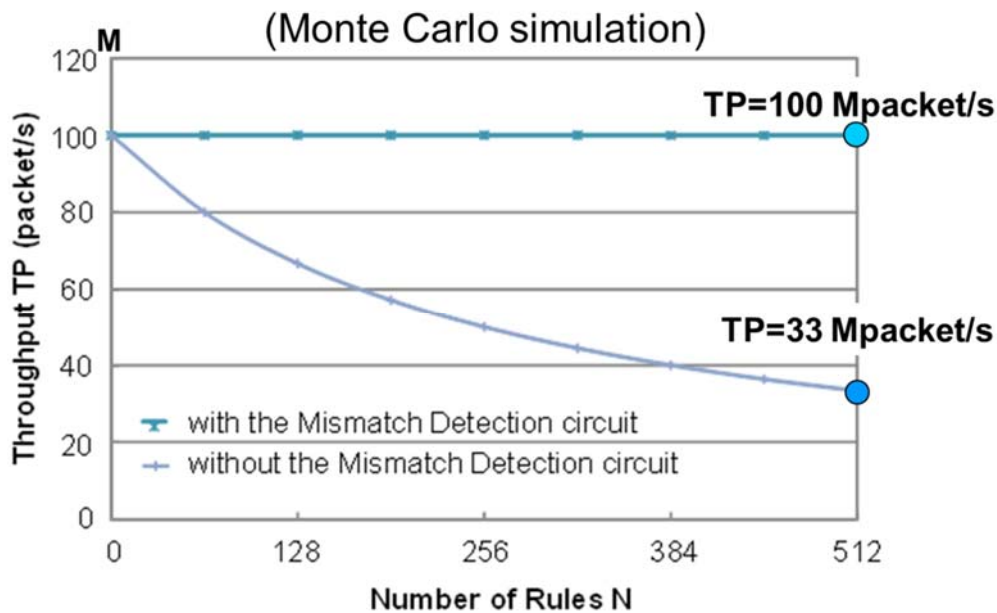


図 6-5 ルール数に対する TP の依存性

また，不一致テーブルの最適な数を調べるため，ルール数と不一致テーブル数を変化させシミュレーションを行った結果を図 6-6 に示す．今回は一致条件/照合データのビット長を 512 ビットとし，分割するビット長を 8 ビットにしたので不一致テーブルの数は 64 個としている．不一致テーブル利用すれば 2 個でも TP が向上する効果が確認できた．また，目標 $TP=80M$ packet/s に合わせた不一致テーブル数にするには 16 個あれば良い．すなわち $N=512$ として 16 個の不一致テーブルを用いた場合の TP は

80.7Mpacket/secであった。また、不一致検出回路を無効にした場合 (N=512), 33.3Mpacket/secであった。したがって不一致検出を組み合わせることで約2.4倍の高速化が図れることがわかった。

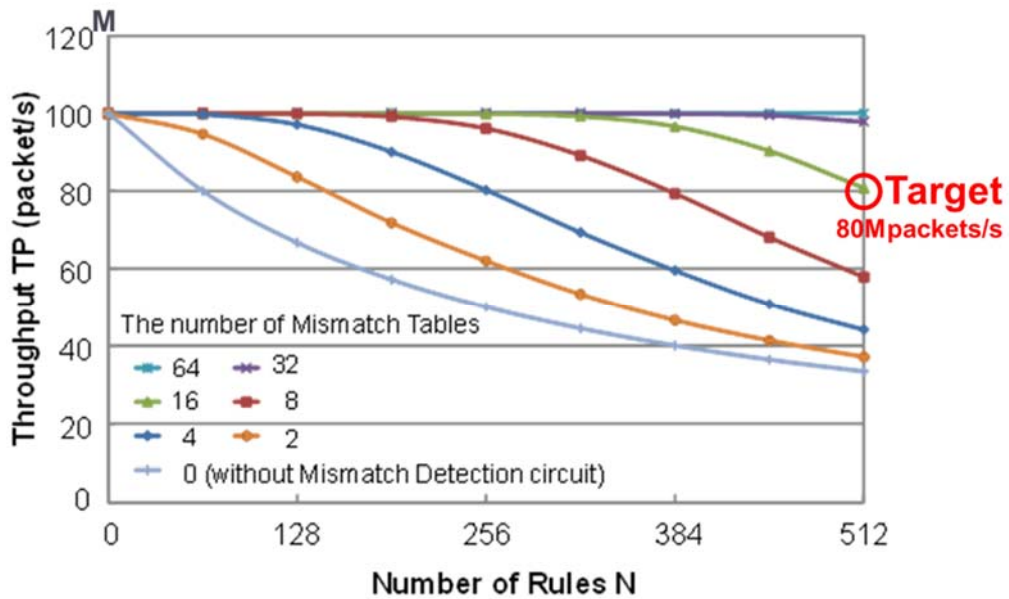


図 6-6 ルール数, 不一致テーブル数に対する TP の依存性

6.5 TEG チップと評価結果

提案したパケット検索エンジンの TEG チップを 40nm 8層メタル CMOS プロセスで試作を行った。図 6-7 に TEG チップ写真およびレイアウトプロット図を示す。また、表 6-4 に TEG チップの諸元を示す。ルールの登録数は 512, 不一致テーブルは 512 ビットルールを 8 ビット毎, 64 のセグメントに分割した, 64 個である。VDD = 1.1V の電源電圧で, 100MHz で動作することを確認した。この時, TP は 100 Mpacket/s (=51.2 Gb / s) である。

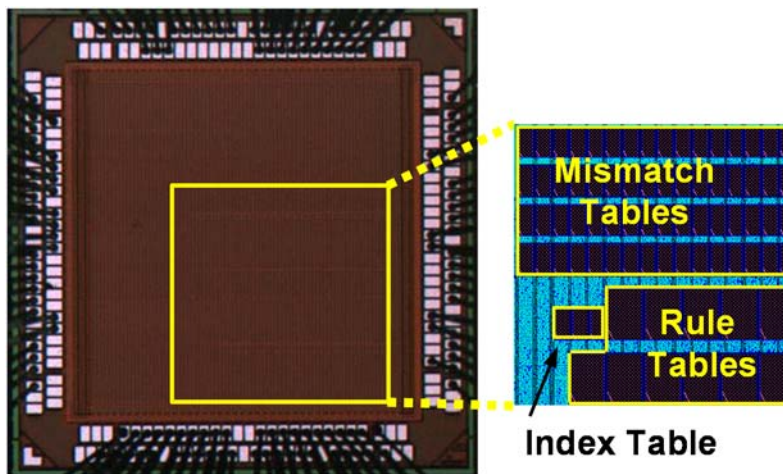


図 6-7 TEG チップ写真とレイアウトプロット図

表 6-4 TEG チップの諸元

Technology	40nm 8 Metal CMOS process
Core size	1,040 μ m \times 1,050 μ m
Supply voltage	1.1 V
Frequency	100 MHz
Registration Rules	512 rules
Rule Length	512 bits
Throughput	100 M packet/s (Target 80M packet/s)
Energy Dissipation	0.808 nJ/Search

コアサイズは1040um \times 1050umである。必要なメモリは、不一致テーブルで256ワード \times 1ビットが64個、索引テーブルで256ワード \times 10ビット、ルールテーブルでは512ワード \times 523ビットである。ただし、今回は、ライブラリの制限から表6-5のような構成になっている。

表 6-5 実装メモリの構成

Table	Required Memory	Implemented Memory
Mismatch	256 words \times 1bit \times 64	256 words \times 16 bits \times 64
Index	256 words \times 10 bits	256 words \times 16 bits
Rule	512 words \times 523 bits	256 words \times 16 bits \times 2 512 words \times 48 bits \times 11

不一致メモリは256ワード \times 16ビットのRAMを64個、索引テーブルは56ワード \times 10ビットのRAMを1個で構成している。ルールテーブルはVB, NF, NAフィールドとFLTフィールドを別のRAMで構成し、それぞれ、256ワード \times 16ビットを2個、512ワード \times 48ビットのRAMを11個使用した。実装したRAMの合計は523Kビットで、必要なRAMの合計280Kビットに対して1.9倍となった。ロジック規模は2入力NAND換算で11.6Kゲートである。

消費エネルギーの測定結果を図6-8に示す。判定サイクル数が既知である照合データを入力し、連続で判定動作を行わせて消費エネルギーEを測定した。

一致検出のみの場合、判定サイクル数が1のときの消費エネルギーはE=0.455nJ/packetであった。また、不一致検出回路利用で判定できた場合、判定サイクル数が1のときの消費エネルギーはE=0.808nJ/packet (ビットあたり1.58 pJ/b \cdot search)であり、この内、不一致検出回路の消費エネルギーは0.353nJ/Searchである。この消費エネルギーは一致検出回路のみの場合に比べ1.77倍となる。ここで、不一致検出回路で判定できた場合の消費エネルギーE=0.808nJに注目する。不一致検出回路が無効の場合、E=0.808nJになるMCは3.32cycleである。MCは一致条件登録数N(ルール数)に依存するので、N(ルール数)が595以上あればMCが3.32以上となり、不一致検出回路を有効にした場合の方が、消費エネルギーが削減できる。

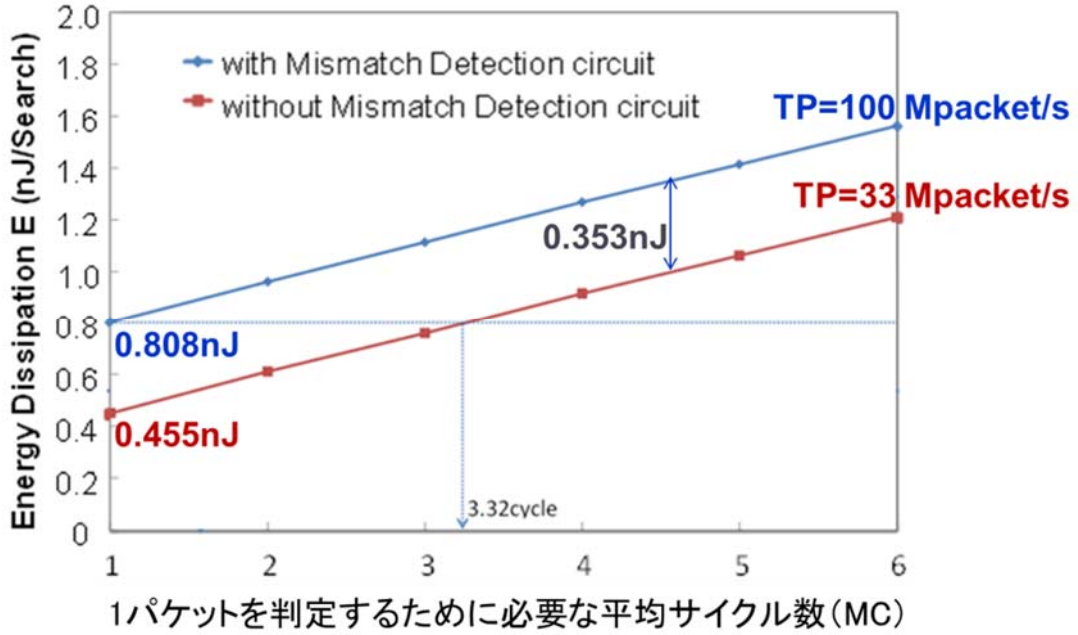


図 6-8 TEG チップの消費エネルギーの測定結果

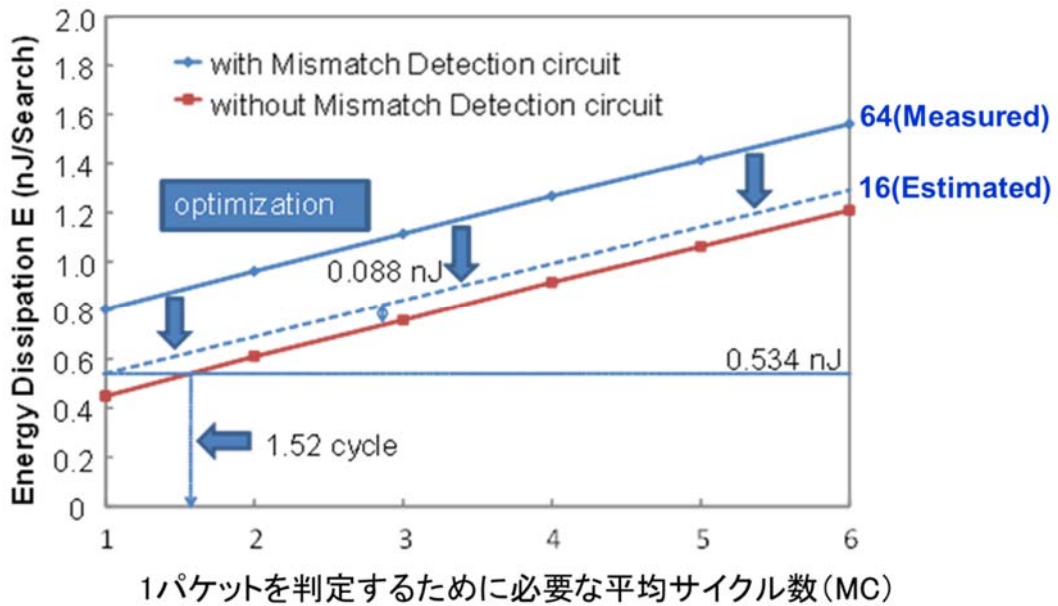


図 6-9 不一致検出回路の最適化による消費エネルギー変化予想

図 6-6 から、不一致テーブル数を 64 個から 16 個に減らし、メモリ容量も最適化した場合、不一致検出回路の消費エネルギーは 1/4 に低減可能である。図 6-9 に不一致検出回路の最適化による消費エネルギー変化予想を示す。不一致検出回路の消費エネルギーが 0.088nJ/Search となり、不一致検出できた場合は 0.534nJ/Search となる見込みを得た。また、その値と不一致検出回路無効時の値が等しくなるのは MC=1.52cycle である。つまり一致条件登録数 $N \geq 134$ であるときに消費エネルギーが改善される。不一致

テーブルを 16 個に減らし、 $N=512$ とした場合でも、不一致検出回路を利用する事で、一致検出回路のみの TP=33.3 Mpacket/sec の 2.4 倍の TP 目標値 80Mpacket/s が実現可能である事が確認できた。

6.6 結言

不一致検出回路とハッシュ探索機能を組み合わせた一致検出回路の新しいパケット検索エンジンを提案し、40nm CMOS プロセスを使って TEG チップを試作した。一致検出回路には 512 ビットの一致条件を最大 512 個登録できるルールテーブル 1 個、不一致検出回路の不一致テーブル数は 64 個とした。実装メモリは 523K ビット、制御ロジック部の回路規模は 2 入力 NAND 換算で 11.6K ゲートである。コア面積は $1040\mu\text{m} \times 1050\mu\text{m} = 1.092\text{mm}^2$ 、電源電圧 1.1V において動作周波数 100MHz を確認した。最大 TP は 100Mpacket/sec (51.2Gbps) である。

平均 TP を、RTL モデルでシミュレーションで測定し、一致条件登録数に関わらず、ほぼ全ての照合データが不一致検出回路で不一致判定できることが確認できた。512 個の一致条件を登録して不一致テーブルの数を 16 個にすると、TP が 80.7Mpacket/sec (目標 80Mpacket/sec \approx 40Gbps) の場合、ルールテーブル数を 64 個から 16 個に削減可能である事がわかった。その場合の不一致検出回路を用いない場合の TP は 33.3 Mpacket/sec であり、約 2.4 倍に高速化する事が確認でき、ハッシュ探索との組み合わせにおいても有効性が確認できた。

また、試作した TEG チップの判定動作の消費エネルギーを測定し、不一致検出回路で不一致判定できた場合、0.808nJ/Search となった。そのうち不一致検出回路の消費エネルギーは 0.353nJ/Search である。また、不一致テーブル数を 64 個から 16 個に最適化する事で、消費エネルギーが 0.534nJ/Search に削減できる見込みが得られた。一致判定回路のみで判定を行った場合と比べると、一致条件登録数が 134 個以上の場合、消費エネルギーが改善できる見込みが得られた。以上より、不一致検出をハッシュ探索に組み合わせることで、TP が向上することを確認し、消費エネルギーも削減できる見込みが得られ、不一致検出回路の有効性を示した。

参考文献

- [6-1] D. E. Taylor, "Survey and Taxonomy of Packet Classification Techniques," ACM Computing Surveys, vol. 37, no. 3, pp. 238-275, Sept. 2005.
- [6-2] D. Knuth, "The Art of Computer Programming, Volume 3: Sorting and Searching".
- [6-3] D. R. Morrison, "PATRICIA-practical algorithm to retrieve information coded in alpha-numeric," J. ACM, Vol.15, No. 4, pp.514-534, Oct.1968.
- [6-4] B. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," Communications of the ACM, vol. 13, no. 7, pp. 422-426, Jul. 1970.
- [6-5] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, J. Lockwood, "Deep Packet Inspection using Parallel Bloom Filters," MICRO, vol. 24, no. 1, pp. 52-61, Mar. 2004.
- [6-6] Young H. Cho, William Henry Mangione Smith, "Deep network packet filter design for reconfigurable devices," ACM TECS, vol. 7, no. 2, article 21, pp. 21:1-21:26, Feb. 2008.

- [6-7] David E. Taylor, "Survey and Taxonomy of Packet Classification Techniques," in *Journal, ACM Computing Surveys*, vol. 37, Issue. 3, Sept. 2005.
- [6-8] N. Miura, S. Shigematsu, T. Hatano, M. Nakanishi, and M. Urano, "A Mismatch-Detection Circuit for a High-Speed Packet Filter," *Proceedings, ITC-CSCC*, pp. 363-366, Jul. 2010.
- [6-9] N. Miura, R. Honda, S. Shigematsu, N. Tanaka, S. Hatta, M. Nakanishi, Y. Matsuda, and M. Urano, "A 96.5% Energy-Reduced Lookup Engine with an Unused-Rules-Exception Scheme for Greening Networks," *Symp. VLSI Circuits, Dig. Tech. Papers*, pp. C288-289, Jun. 2013.
- [6-10] <http://www.ieee802.org/3/ba/index.html> "IEEE P802.3ba 40Gb/s and 100Gb/s Ethernet Task Force,"

第7章 結論

本研究は、多くのマイコン製品ファミリー、派生品種数を抱える半導体メーカーが、今後迎える少量多品種時代を見据え、マイコン製品の品種削減（コスト削減）を行うことを目的とし、内蔵メモリとして利用、かつマイコン周辺回路を再構成可能とするマイコン周辺回路向けに特化したプログラマブルロジックデバイス FPSM アーキテクチャの提案を行った。従来のプログラマブルデバイスのユーザであるハードウェアエンジニアだけでなく、ソフトウェアエンジニアも対象ユーザとするメモリをベースとした新しいコンセプトのフィールドプログラマブルデバイスの提案を行なった。その成果を以下にまとめる。

7.1 基本論理素子 PMU アーキテクチャ

新しい基本論理素子 PMU は、粗粒度のメモリを利用し、CPU のプログラムカウンタと同様にマイクロ命令を使ったアドレス制御によるシーケンシャル動作、条件分岐/無条件分岐命令を実行し、プログラムされた動作とメモリ出力を行うことで、特定の機能を実現（順序回路を模擬）できる。この PMU アーキテクチャを、SystemC を用いたモデルベース開発手法を用いて開発した。シミュレーションモデル上に回路機能を実現する真理値表とマイクロ命令を実装し、シミュレーションによる動作確認、波形観測を行とともに、PMU アーキテクチャの改良とそのマイクロ命令定義・体系化を行い、基本論理素子として利用可能であることを確認した。

7.2 FPSM アーキテクチャ

メモリをベースとした基本論理素子 PMU と、これらを複数つなげて利用するためのスイッチ機構 (SB) を組み合わせアレイ構成にするとともに、マイコンに搭載するための MCU インタフェースを追加し、FPSM アーキテクチャを開発した。また、これらにマイコンで多用される基本的な論理演算回路であるカウンタ/タイマ、シフトレジスタ、加算器およびキャプチャ機能等をシミュレーションモデル上に実装し、動作確認を行った。さらにマイコンの周辺回路として FIFO 機能、シリアル通信インタフェース機能および PWM 機能をシミュレーションモデルに実装し、シミュレーション波形観測による動作確認を行い、実装したマイコン周辺回路機能が設計通り動作していることを確認した。

また、このマイコン周辺回路の中から 8 ビット PWM を RTL 設計し、FPGA 上で実装評価を行った。市販の ALTERA 社製 FPGA ボードを用い、3 個の PMU を使って 8 ビット PWM を実装した。実装には同社から提供される QuartusII Ver6.1 を使用し、論理合成を行った。今回は SB 無しの実装であったがシミュレーションモデル同様、設計通りの結果であった。以上により、モデルベース開発が有効な手段であることも確認できた。さらに、今回提案した FPSM アーキテクチャを、 $0.18\mu\text{m}$ CMOS プロセスを用い、PMU 4×4 アレイ構成の FPSM 実験チップを論理設計・実装設計および評価を行った。ゲート規模は 2 入力 NAND 換算で 46k ゲート、コア部の面積は 2.265mm^2 であった。カウンタ/タイマ、シフトレジスタ、シリアル I/O、FIFO、PWM 等、想定した周辺回路機能が全て実験チップ上に実装し、再構成できることを確認した。消費電力は、電源電圧 1.8V、動作周波数 50MHz において、約 1mW/PMU が得られた。これらの結果から、ALTERA 社 Stratix II シリーズの FPGA と実装面積と消費電力の比較考察を行い、FPSM は FPGA の実装面積の同等か半分以

下、消費電力はFPGAの約1/3から1/5程度の低消費電力化が可能であることがわかった。特定のマイコン周辺回路という限定的な回路実装の比較ではあるが、実装面積および消費電力ともにFPGAよりもFPSMアーキテクチャの方が優位であり、マイコン製品に搭載可能なプログラマブルロジックデバイスとして十分利用可能である見通しを得た。

7.3 応用展開

基本論理素子PMUを応用したパケット検索エンジンの研究を行った。一致検出回路に用いるハッシュテーブルには、片方向のリンクリストを用いた複数のハッシュグループが実装される。通常これらはプログラムで実装されている。PMUでは自律シーケンス動作が可能であり、PMUを利用することでハードウェア上にハッシュ機能を実現している。

一致/不一致検索エンジンにおいて、高速なハッシュ探索回路を一致検出回路に実装するとともに、不一致検出回路を組み合わせることで高スループット、かつ低消費電力なパケットフィルタ回路を提案し、40nm 8層メタルCMOSプロセスを用いてTEGチップを試作した。

試作したTEGチップは、電源電圧1.1V、動作周波数100MHzにおいて、ルール長512ビット、登録ルール数512個で、100Mpacket/s(=51.2Gbps)のスループットを実現している。また、この時の消費エネルギーは0.808nJ/Searchであり、不一致検出とハッシュ探索を組み合わせることで、スループットが向上することを確認し、不一致テーブルの最適化設計により、消費エネルギーも削減できる見通しが得られ、有効性を示した。

7.4 今後の課題と展望

冒頭でも述べたが、半導体の少量多品種の時代の到来が予測される中、マイコンなどの多品種製品の少量生産は半導体メーカーにとっては、コスト削減の流れと逆行している。さらに、これまでのマイコンビジネスの継続で、過去からの製品も含め、既に少量多品種ビジネスに陥っている。これからの少量多品種の時代の到来にどう立ち向かうかで、マイコンのコスト競争で生き残れるかが決まる。既に数社はPLDを搭載するマイコンを製品化しているが、まだ一般化はしていない。これらは、RTL設計などのハードウェア設計スキルが求められるが、FPSMはハードウェア設計スキルを必要としないソフトウェアエンジニアも対象ユーザとした新しいコンセプトのメモリベースのフィールドプログラマブルロジックデバイスである。このため、実装方法もソフトウェアエンジニア向けの新しい手法が必要になる。現在は、本研究で利用した真理値表とSBの結線情報をライブラリ化して実装する手法を準備しているが、最終的にはこれら真理値表とSBの結線情報を各要素に分割/分類して、共通のライブラリ化とリンク情報を準備し、抽象化またはグラフィカルなユーザインタフェースを用いたグラフィカルユーザインタフェース等の実装方法の提供が必要と考える。

また、現状のFPSMでは、基本論理素子PMUのメモリサイズは4Kビットであり、PMUを結線するためのSBのロジック部の冗長比率が高い。さらに第4章でも述べたが、PMUアレイ構成でPMUを複数結線・配置する場合、実装する周辺回路機能によって、本来PMUが4個以内であれば、1行で収まる予定であったが、利用するフラグ出力の都合で、PMUが4個以内でも2行にわたって配置する必要が出てきた。これに

より FPSM の実装効率が悪化する。今後マイコンに搭載する場合は、FPSM の出力段を工夫するとともに、さらに効率の良いメモリの利用方法、かつプログラマブルロジックデバイスとしての利用するためのメモリの粒度やSBのスイッチ回路を簡略化など検討も必要と考える。

FPSM はハードウェア設計スキルを必要としないマイコン周辺回路に特化した新しいプログラマブルロジックデバイスであり、さらに製品化に向けた冗長部分の簡略化や利用方法の開拓などの課題へ取り組むとともに、将来、FeRAM やMRAM 等の不揮発性RAM 技術を利用することで、フラッシュROM によるソフトウェアのフィールドプログラマブルだけでなく、ハードウェアのフィールドプログラマブルの利用も可能な新しいビジネスモデル創生や新しいプラットフォーム開発が期待される。

謝辞

本論文をまとめるにあたり、終始暖かい激励とご指導、ご鞭撻を頂いた金沢大学理工研究域電子情報学系 松田吉雄教授に深甚なる謝意を表します。また、多くの有益なる御教示と御指導を賜りました金沢大学理工研究域電子情報学系 今村幸祐准教授、深山正幸講師、ならびに秋田純一教授、北川章夫教授に深謝の意を表します。

本論文は、筆者が平成 17 年から平成 25 年の間にルネサスエレクトロニクス株式会社および金沢大学在学中の研究をまとめたものである。本研究にあたって、終始ご指導とご討論を頂きました日本大学工学部情報工学科（元ルネサスエレクトロニクス株式会社）松村哲哉教授に感謝の意を表します。また、本研究の機会を与えて頂いた名古屋電機工業株式会社 ITS 情報装置事業本部グローバル事業推進室室長（元ルネサスエレクトロニクス株式会社）坪井務氏、岡山県立大学（元ルネサスエレクトロニクス株式会社）有本和民教授に感謝の意を表します。

本研究を進めるにあたりご協力および有益なご討論を頂きました株式会社日立産業制御ソリューションズ組み込みソリューション本部主任技師 梶原久志氏、ルネサスエレクトロニクス株式会社第一ソリューション事業本部コア技術事業統括部 土屋浩氏に感謝いたします。

業績目録

1. 学術論文 (査読有)

- [1] T. Sasao, H. Nakahara, M. Matsuura, Y. Kawamura, and J.T. Butler, "A quaternary decision diagram machine: Optimization of its code," IEICE Transactions on Information and Systems, Vol. E93-D, No.8, pp.2026-2035, Aug. 2010.
- [2] H. Nakahara, T. Sasao, M. Matsuura, and Y. Kawamura, "A parallel branching program machine for sequential circuits: Implementation and evaluation," IEICE Transactions on Information and Systems, Vol. E93-D, No.8, pp. 2048-2058, Aug. 2010.
- [3] Y. Kawamura, N. Okada, Y. Matsuda, T. Matsumura, H. Makino, and K. Arimoto, "A Field Programmable Sequencer and Memory with Middle Grained Programmability Optimized for MCU Peripherals," IEICE Trans. Fundamentals, Vol. E99-A, No.5, pp.917-928, May.2016.

2. 国際学会 (査読有)

- [1] Y. Kawamura, "A reconfigurable microcomputer system with PA³ (Programmable Autonomous Address-control-memory Architecture)," IEEE Asian Solid-State Circuits Conference, Proceedings, pp. 388-391, Nov. 2007.
- [2] H. Nakahara, T. Sasao, M. Matsuura, and Y. Kawamura, "Emulation of sequential circuits by a parallel branching program machine," 5th International Workshop on Applied Reconfigurable Computing, Lecture Notes in Computer Science 5443, pp. 261-267, Mar. 2009.
- [3] T. Sasao, H. Nakahara, K. Matsuura, Y. Kawamura, and J.T. Butler, "A quaternary decision diagram machine and the optimization of its code," 39th International Symposium on Multiple-Valued Logic 2009, May 2009.
- [4] H. Nakahara, T. Sasao, M. Matsuura, and Y. Kawamura, "The parallel sieve method for the virus scanning engine," 12th Euromicro Conference on Digital System Design 2009, pp. 809-816, Aug. 2009.
- [5] T. Matsumura, K. Imamura, Y. Kawamura, and Y. Matsuda, "Automatic Rule Registration and Deletion Function on a Packet Lookup Engine LSI," The 2016 International Symposium on Intelligent Signal Processing and Communication Systems, Proceedings, pp. 34-39, Oct. 2016.
- [6] H. Nakahara, T. Sasao, M. Matsuura, and Y. Kawamura, "A virus scanning engine using a parallel finite-input memory machine and MPUs," International Conference on Field-Programmable Logic 2009, Sep. 2009.
- [7] H. Nakahara, T. Sasao, M. Matsuura, and Y. Kawamura, "PBM128: A parallel branching program machine consisting of 128 branching program machines," 19th International Workshop on Post-Binary ULSI Systems, May 2010.
- [8] T. Matsumura, N. Okada, Y. Kawamura, K. Nii, K. Arimoto, H. Makino, and Y. Matsuda, "The LSI Implementation of a Memory Based Field Programmable Device for MCU Peripherals," IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, Proceedings, pp. 183-188, Apr. 2014.

- [9] Y. Kawamura, K. Imamura, N. Miura, M. Urano, S. Shigematsu, and Y. Matsuda, “A 100-MHz 51.2-Gb/s packet lookup engine LSI based on mismatch detection circuit combined with linked-list hash table,” The 2015 International Symposium on Intelligent Signal Processing and Communication Systems, Proceedings, pp. 351-356, Nov. 2015.

3. 国内学会・研究会等（査読無）

- [1] 中原啓貴, 笹尾勤, 松浦宗寛, 川村嘉郁, “並列ブランディング・プログラム・マシンを用いた順序回路の模擬について,” 信学技報 VLD2009-145, pp.111-116, 2009年3月.
- [2] 中原啓貴, 笹尾勤, 松浦宗寛, 川村嘉郁, “並列ふるい法と MPU を用いたウイルス検出エンジンについて,” 信学技報 RECONF2009-45, pp.25-30, 2009年12月.
- [3] 中原啓貴, 笹尾勤, 松浦宗寛, 川村嘉郁, “並列ブランディング・プログラム・マシンを用いたパケット分類器について,” 信学技報 Reconf2009-77, pp.143-148, 2010年1月.
- [4] 松村哲哉, 川村嘉郁, 岡田尚也, 有本和民, 牧野博之, 松田吉雄, “メモリをベースにした MCU 内蔵省電力プログラマブルデバイス,” 信学技報 VLD2013-46, pp.1-6, 2013年10月.
- [5] 川村嘉郁, 岡田尚也, 松田吉雄, 松村哲哉, 牧野博之, 有本和民, “メモリをベースにしたマイコン周辺回路用フィールドプログラマブルデバイスの LSI 実装,” 信学技報 VLD2014-108, pp.239-234, 2014年11月.
- [6] 川村嘉郁, 今村幸祐, 三浦直樹, 浦野正美, 重松智志, 松村哲哉, 松田吉雄, “不一致検出とハッシュ探索に基づくパケット検索エンジン LSI,” 信学技報 VLD2015-118, pp.43-48, 2016年2月.

4. 知的財産

- 1) 川村嘉郁, 特許第 4852149 号, “半導体装置,” 2011年10月.
- 2) 笹尾勤, 中原啓貴, 川村嘉郁, 特許第 5382503 号, “ブランディング プログラム マシン及び並列プロセッサ,” 2013年10月.
- 3) 川村嘉郁, 中野裕文, 河合浩行, 特許第 5560463 号, “半導体装置,” 2014年6月.