令和 5 年度

修士学位論文

データ駆動型プロセッサの環状パイプライン構成法

A Study on Circular Pipeline Configuration for Data-Driven Processors

1265106　　髙橋　龍一

指導教員　　情報学群 岩田誠

2024 年 2 月 28 日

高知工科大学大学院 工学研究科 基盤工学専攻

情報学コース

要 旨

データ駆動型プロセッサの環状パイプライン構成法

髙橋　龍一

近年，IoT(Internet of Things) 技術の普及により，IoT 機器は幅広い分野で利用されている．それに伴い IoT デバイスはより高性能かつより省電力であることが求められている．そのため，局所動作により，高性能性と省電力性を実現する STP(Self-Timed Pipeline) が考えられた．この STP を利用し，非同期で回路が動作し，多重並列処理が可能であるデータ駆動型プロセッサ DDP が研究されている．DDP ではプログラム実行に必要な構成要素を環状に接続したパイプラインで構成されており，その構成要素の接続順により回路規模，性能，パケット構成，および，メモリ量に差が出ることが明らかとなっている．

本研究ではパイプラインの構成における，構成要素の最適な接続順序について検討し，より有望であると考えられる PS 先行型 DDP 構成を設計した．また，DDP の構成要素の一つである，マッチングメモリは，BRAM を使用した RAM 型とレジスタを使用したレジスタ型の 2 種を実装し，それぞれ比較・評価を行っている．本研究では，IoT 向き DDP 使用を対象として，従来のパイプライン構成である PS 後続型 DDP 構成と本研究で提案する PS 先行型 DDP 構成とを FPGA 回路設計して比較評価を行った．その際，構成要素間で授受されるパケット流量を明らかにし，その簡易モデルから各構成要素の稼働率を導き出している．

結果，RAM 型では LUT 数は 31%の削減，FF は 24%の削減に成功し，ジスタ型では LUT 数は 10%の削減，FF は 0.06%の削減に成功した．また，本研究で提案する PS 先行型 DDP 構成は，回路規模削減率と稼働率から各パイプライン構成の総消費電力削減率を概算した結果，4 つの極端なプログラムモデルを使用し，いかなる場合でも PS 先行

型 DDP 構成は総消費電力量を削減できた.

キーワード　データ駆動型プロセッサ，パイプライン構成，省電力化，パケット流量，モデル化

Abstract

A Study on Circular Pipeline Configuration for Data-Driven Processors

Ryuichi TAKAHASHI

In recent years, IoT (Internet of Things) technology has become widespread, and IoT devices are used in a wide range of fields. IoT devices are required to have higher performance and lower power consumption. Therefore, STP (Self-Timed Pipeline) was considered because it can achieve high performance and low power consumption through localized operation. The DDP is a data-driven processor that operates asynchronously and is capable of multiple parallel processing. The order in which the components are connected has been shown to cause differences in circuit size, performance, packet composition, and amount of memory. In this study, we investigated the optimal connection order of the components in a pipeline configuration and designed a PS-first DDP configuration because of its more promising prospect. In addition, we implemented two types of matching memory, a RAM type using BRAM and a register type using registers and compared and evaluated each type of memory. In our research, we designed and evaluated FPGA circuits for the use of DDP for the IoT and compared the conventional pipeline configuration (PS subsequent type DDP configuration) and the PS preceding type DDP configuration proposed in this study. The packet flow rate exchanged between the components is clarified, and the utilization rate of each component is derived from the simplified model. As a result, the RAM type succeeded in reducing the number of LUTs by 31% and the FF by 24%, while the GISTA type succeeded in reducing the

number of LUTs by 10% and the FF by 0.06%. In addition, we estimated the total power reduction for each pipeline configuration based on the circuit size reduction ratio and utilization ratio, and we found that the PS-precedence DDP configuration proposed in this study reduced the total power consumption in all cases using the four extreme program models.

key words data-driven processors, pipeline configuration, lowpower consumption, acket flow, modeling

目次

図目次

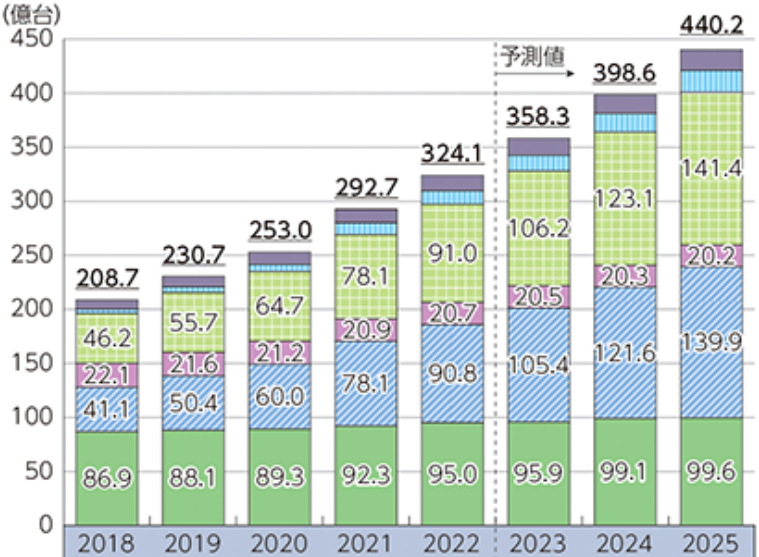
図目次

表目次

第 1 章

序論

近年，IoT(Internet of Things) 技術の普及により，IoT 機器は幅広い分野で利用されている．従来のインターネット接続端末以外にも，家電や自動車，ひいては，ビルや工場など様々なものがネットワークに繋がる時代となっている．その中でも，図 1.1 に示すように，「コンシューマ」，「産業用途」は急激に増加している．また，「医療」，「自動車・宇宙航空」業界において，IoT 機器の利用が今後急激に成長すると予想されている [1].



	2018	2019	2020	2021	2022	2023	2024	2025
合計	208.7	230.7	253.0	292.7	324.1	358.3	398.6	440.2
自動車・宇宙航空	8.5	9.8	11.3	12.7	14.2	15.7	17.3	18.9
医療	4.0	5.1	6.5	10.7	12.5	14.6	17.2	20.2
産業用途	46.2	55.7	64.7	78.1	91.0	106.2	123.1	141.4
コンピュータ	22.1	21.6	21.2	20.9	20.7	20.5	20.3	20.2
コンシューマ	41.1	50.4	60.0	78.1	90.8	105.4	121.6	139.9
通信	86.9	88.1	89.3	92.3	95.0	95.9	99.1	99.6

(出典)Omdia

図 1.1: 世界の IoT 機器の数と推移及びその予想 (文献 [2] より引用)

それに伴い，IoT デバイスは省電力化と高性能化の需要が高まっている．IoT デバイスは，主にエッジコンピューティングやフォグコンピューティング機器として用いられている．その際，センシングされた大量のデータをより高速に処理する能力 (高性能化) が求められる．また，IoT デバイスはいかなる場合でも常に電力を確保できるとは限らない [3]，さらに，電力供給型でなくバッテリーで動作する IoT デバイスも存在する．それゆえ，センサーがより長時間データを取得し続ける能力 (省電力化) は IoT デバイスが満たすべき重要な要件の一つである [4]．省電力を実現するためには，動的及び静的な消費電力の削減が有効である．動的な消費電力は，データを処理する際の不要なトランジスタのスイッチングや必要処理速度よりも高い過剰なスイッチングが原因である．また，静的な消費電力は非活性トランジスタを通るリーク電流が主な原因である [5]．その対策として，静的なリーク電流による損失を防ぐために，デジタル回路の田ある供給制御技術であるダイナミック電圧スケーリング (DVS) やパワーゲーティング (PG) 等の技術が考えられている [6]．また，大域的クロックを用いないため，動作する必要のない構成要素ではクロック信号が供給されず，動的な電力が無駄に消費されることのない省電力な回路である，セルフタイム型パイプライン STP(Self-Timed Pipeline) が考えられた．本論文では，データラッチ DL，ロジック Logic，データ転送制御素子 C を 1 つの構成要素とし設計されている．これらの技術を用いることで，パイプライン段単位に，PG 用回路を個別に追加することによりパイプライン間で授受されるデータ転送制御素子の信号からパイプライン段ごとに電力の供給・遮断を制御することが可能となる [7].

本研究では STP の技術を利用し，非同期で回路が動作し，多重並列処理が可能なデータ駆動プロセッサ DDP(Data-Driven Processor) を実現する技術を検討している．DDP はプログラムの実行制御に必要な構成要素を環状に接続したパイプラインで構成されている．また，構成要素間で授受されるデータにはパケット形式が採用されている．一方で，これらの構成要素はその接続順序により，各構成要素の回路，パケット構成，および，必要なメモリ量が異なり，結果として，DDP の実装に必要な回路規模や性能が異なることが分かっている [10].

本研究では，これらの組み合わせを網羅的に比較検討し，各パケット長の短縮が高性能化や回路規模削減に寄与することを考慮し，提案する DDP 構成をパケット形式を含めて検討している．本論文において，第 2 章では DDP の基本的な構成要素について述べ，また，それらの構成要素を順に処理する環状パイプラインの構成について述べる．第 3 章では，本研究で提案する PS(Program Storage) 先行型 DDP 構成について述べる．第 4 章では，従来の DDP 構成である PS 後続型 DDP 構成と本研究で提案する PS 先行型 DDP 構成を比較し，その評価について述べる．第 5 章では，本研究のまとめと今後の展望について述べる．

第 2 章

DDP のパイプライン構成

2.1 緒言

第 1 章で述べたように DDP は，非同期で回路が動作し，データ駆動処理方式を採用することにより多重並列処理を可能としたプロセッサである．データ駆動処理方式については後ほど詳細を説明する．各構成要素間で，パケットを転送するか否かは，各構成要素ごとに用意されているデータ転送制御回路 (C 素子) によって制御されている．また，DDP は構成要素の接続順序を入れ替えても機能することが明らかとなっている．

よって，本章では DDP の基本的な動作と構成要素について述べ，本研究で提案するパイプライン構成法についての検討を説明する．

2.2 データ駆動処理方式

従来のプロセッサでは逐次型処理方式が広く採用されていた．このデータ処理方式は，命令をプログラムカウンタから読み出し，1 つの命令を実行後に，次の命令を読み出し実行する．そのため，原則プログラムは in-order で実行され，実行する命令によっては命令の処理を終えるまで待機する必要がある．一方，データ駆動処理方式では，データ依存の関係にないデータは多重処理が行える．すなわち，逐次型処理方式ではプログラムを in-order でしか処理することができないのに対し，データ駆動処理方式は多重並列処理が行えるため逐次型処理よりも高い性能と省電力を実現できる．

データ駆動処理方式は前述した通りデータを基に演算を行う．図 2.1 に示すようにプ

ログラムはそれぞれの命令実行に必要なデータが到着すると，発火し実行可能状態になる．実行可能状態になると，各トークンが保持している情報を基に演算が実行される．また，演算された結果はトークンが保持している宛先に従って，次に実行する演算の引数としてトークンの形で出力される．

図 2.2 では具体的な演算例を用いて説明する．今回の例では，$(A+B)-(C+D)$ の命令を実行する場合を考える．この図において，dest は各データの宛先を表している．この演算を行うには，加算 (ADD)，減算 (SUB) の各命令において，データの受け渡しが必要となる．その際，SUB は dest1 と dest2 の演算結果を入力データとして使用するため，dest1 と dest2 との依存関係がある，それに対して，ADD は依存関係がないため，並列処理を行うことができる．その後，dest1 と dest2 が演算を終え，データが次の宛先である dest3 に出力され，また新たな入力データとして受け渡され演算が行われる．

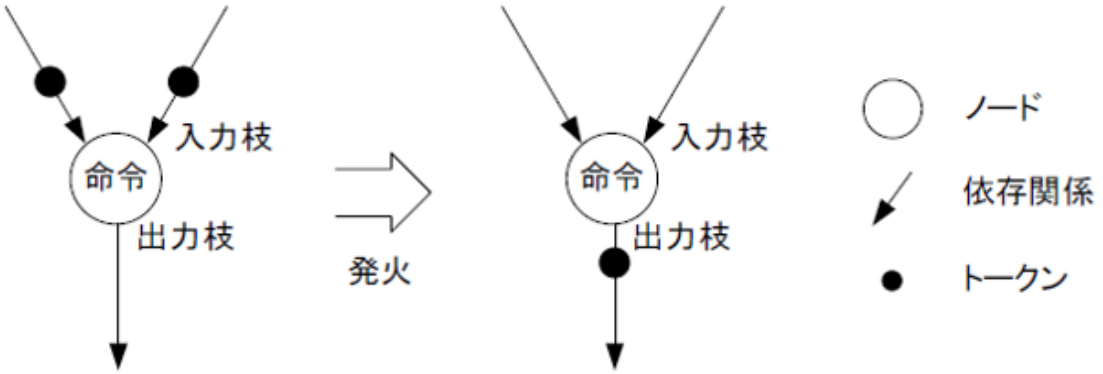


図 2.1: データ駆動の動作原理

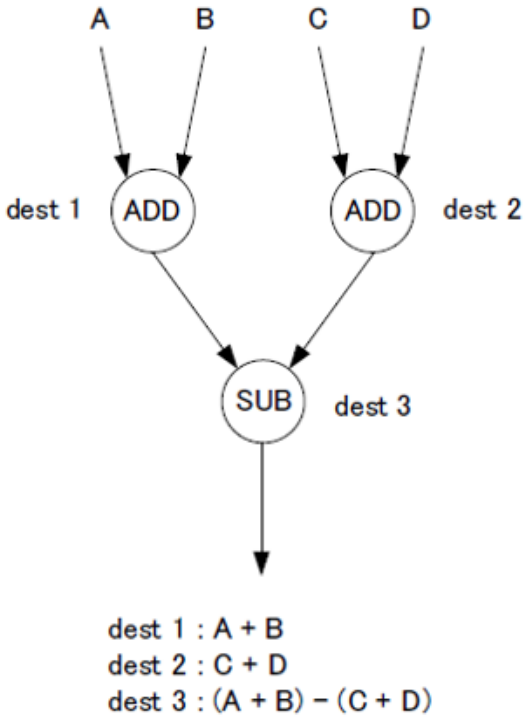


図 2.2: データフローフラフの例

2.3　セルフタイム型パイプライン STP

STP は大域的クロックを用いない省電力なデータ処理を実現する技術である．図 2.3 に示すように STP は，ラッチ回路を任意の個数並べることにより，入力されたデータ情報を保持することが可能なデータラッチ DL，DL で保持された情報を基に演算を行うロジック Logic，DL 間でデータを転送するか否かを制御するデータ転送制御回路によって構成されている．また DL，Logic，データ転送制御回路を一つの Pipeline Stage または構成要素と呼ぶ.

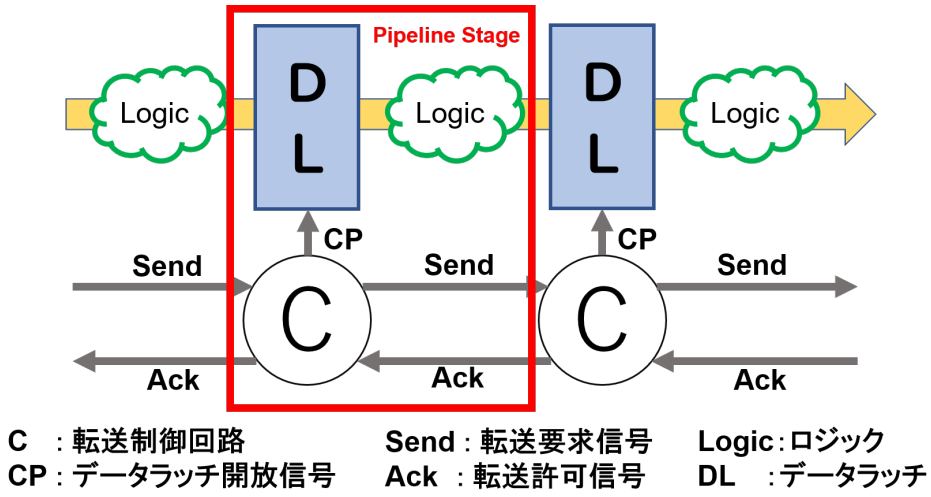


図 2.3: セルフタイム型パイプライン STP(Self-Timed Pipeline)

STP では大域的クロックを用いて同期しない代わりに，C 素子と呼ばれるデータ転送制御回路が用意されている．C 素子がハンドシェイクすることで，データを次の構成要素に転送するか否かを決定している．具体的な C 素子同士のハンドシェイク方法は以下のようになっている [8].

1. データ要求信号が C 素子に入力される
2. C 素子は転送許可信号を前段の C 素子に出力する
3. データ要求信号が再び入力される
4. ラッチ開放信号が出力される

このように前段・後段の C 素子間でデータ転送制御の可否を通信することにより，大域的クロックを用いず各構成要素が独立してデータ処理を行うことができるようになる．すなわち，データ処理を行う構成要素のみが動作するため，省電力化の点において有用である [11].

2.4　データ駆動型プロセッサ DDP

本節では STP を用いて設計された DDP について説明する．既存の DDP 構成である PS 後続型 DDP 構成について図 2.5 に示す．

2.4.1　パケットフォーマット

DDP では演算時に使用されるデータと演算に必要な情報を付与したヘッダをビット結合し，図 2.4 のようにパケット形式でデータの授受を行っている．データには演算を行う際に必要なデータが格納されている．ヘッダではデータの宛先 dest，実行される演算コード OP，また各種フラグが用意されている．

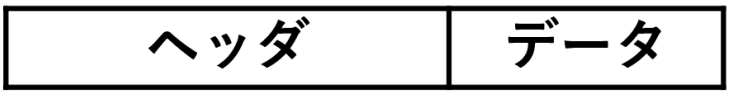


図 2.4: パケット形式

2.4.2　PS 後続型 DDP 構成

図 2.5 で示すように従来の DDP 構成である PS 後続型 DDP 構成は八つの構成要素が環状に接続されたパイプライン構成である．PS 後続型 DDP の各構成要素の詳細について以下に示す．

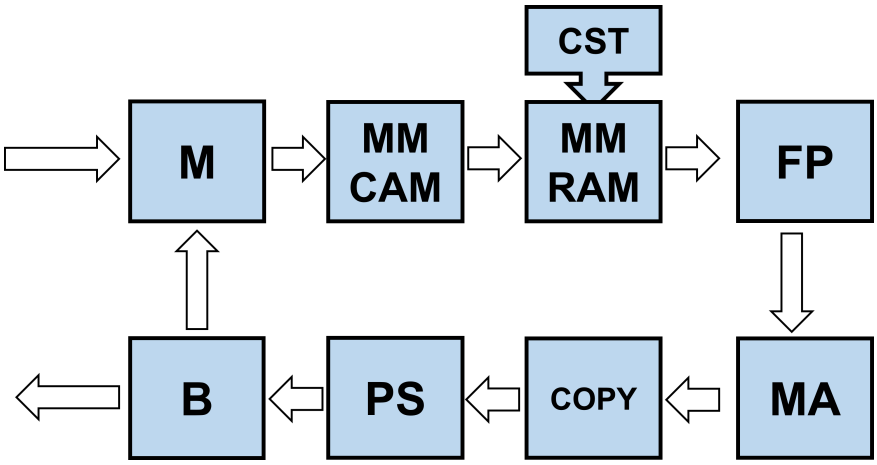


図 2.5: PS 後続型 DDP 構成

- 合流 (Merge Unit:M)

  外部からの入力パケットとパイプライン内を周回してきたパケットとの合流を検出し，適したパケットを出力する．

- パケット待合せ (Matching Memory:MMCAM)

  待合せを行うかどうかを判断する．二項演算である場合は対となるオペランド対を検出する．また，待合せをしているパケットのヘッダ部をマッチングメモリに保持しておく．

- パケット結合 (Matching Memory:MMRAM)

  オペランド対を検出するまで，パケットが待機するためのメモリを用意し，対となるオペランドが到着するまでメモリにパケットを保持しておく．

- 定数読み出し (Constant Memory:CST)

  定数演算を行う際，演算に必要となる定数が定数メモリ CST(Constant Memory) に格納されている．定数演算を行う場合は，パケットが保持している宛先情報 dest をアドレスとして CST にアクセスし，メモリから定数と半分に分割された命令コードを読み出し，パケットに報を付与する．定数演算を行わない場合はそのまま入力パケットが次の構成要素へ出力される．また，この構成要素は MMRAM と結合し 1 つの構成要素となっている．

- 演算 (Functional Processor:FP)

  パケットが保持している演算コード OP を基に，パケットが保持している二つのデータの算術演算や論理演算を行う．また，ロード命令やストア命令などでは二つのデータを基にアクセスするデータメモリへのアドレスを算出している．

- メモリアクセス (Memory Access:MA)

  演算結果を保存，もしくは取り出すためのデータメモリにアクセスする．前段で演算された結果をロードする命令の場合，パケットのデータが保持している情報をアドレスとしてデータメモリにアクセスし，データを取り出し，パケットに付与する．また，ストア命令を実行する場合は，演算時の二つのデータのうち片方がアドレス，もう一方が格納するデータとなっている．そのアドレスを参照して指定された番地に演算結果を格納する．ロード・ストア命令以外の場合は次の構成要素へそのままパケットが出力される．

- パケット複製 (Copy Unit:COPY)

  入力パケットはパケット複製を行うか否かを判断する，CPY というフラグの情報を有している．この構成要素ではそのフラグを基にパケットを複製するか否かの判定を行い，フラグが 1 であればパケット複製後，dest を 1 加算し出力する．パケット複製を行わない場合は入力パケットをそのまま次の構成要素へ出力する．

- 命令フェッチ (Program Storage:PS)

  パケットの宛先を管理する，dest をアドレスとして，PS メモリにアクセスし，パケットの次の宛先，演算コード，パケット複製を行うか否かの情報を更新する．

- 分流 (Branch Unit:B)

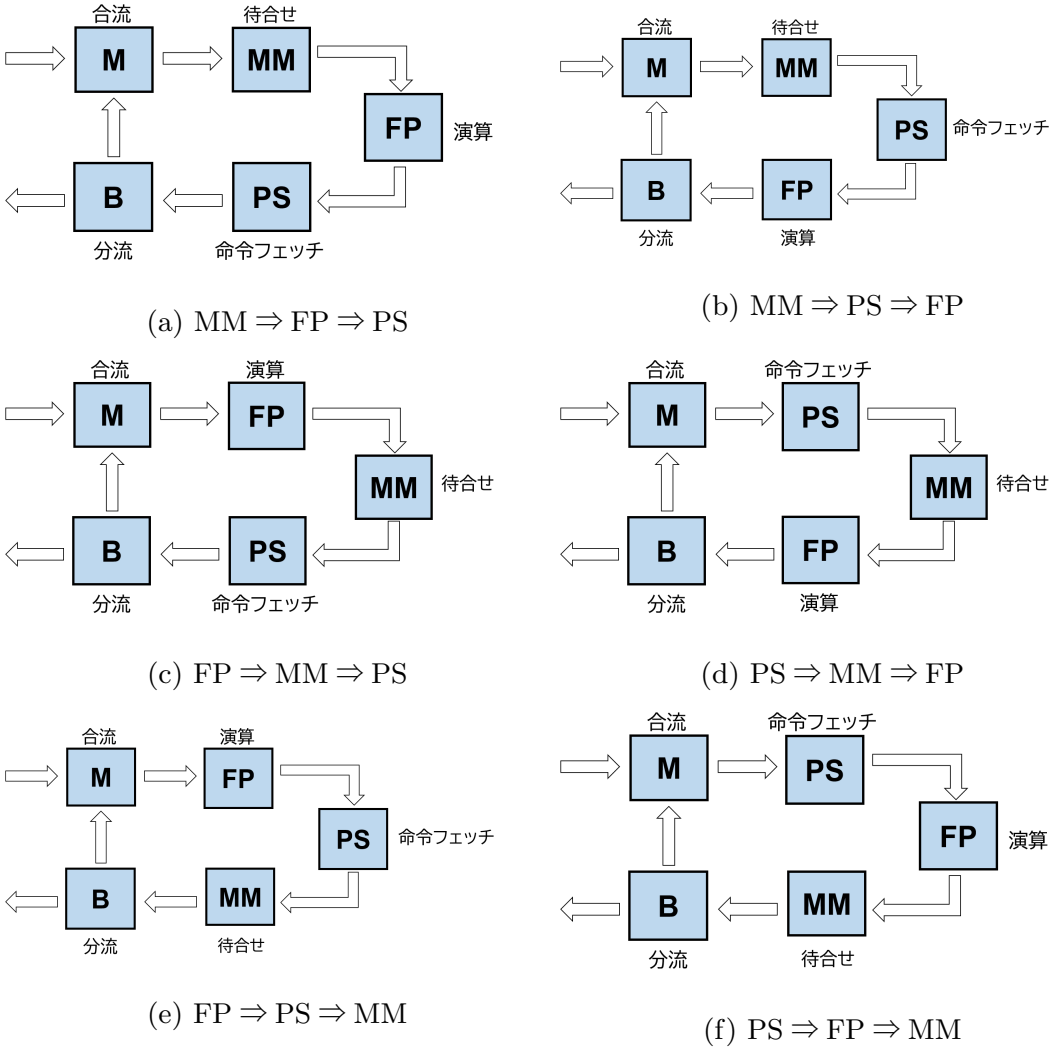
  パケットが保持している，フラグの情報を基として，パケットをパイプライン内で巡させるか，外部に出力させるかを判断し，パケットを出力する．

これらの構成要素のうち，MM(MMCAM&MMRAM)，FP，PS は二項演算を行う際に必要不可欠な構成要素である [11]．そのため本論文ではこれら三つの構成要素を総称し

て基本構成要素と呼ぶ.

2.5　基本構成要素の接続順序

前述したとおり，基本構成要素は DDP に不可欠な構成要素であり，これらの構成要素を順に実行する環状パイプラインは，図 2.6 に示すように 6 通りが考えられる．しかし，基本構成要素の接続順序によって，各構成要素の回路，パケット構成，および，必要なメモリ量が異なり，結果として，DDP 実装に必要な回路規模や性能が異なる．よって，本研究では，これらの組み合わせを網羅的に比較し，構成要素の接続順序について検討している.



(a) MM ⇒ FP ⇒ PS

(b) MM ⇒ PS ⇒ FP

(c) FP ⇒ MM ⇒ PS

(d) PS ⇒ MM ⇒ FP

(e) FP ⇒ PS ⇒ MM

(f) PS ⇒ FP ⇒ MM

図 2.6: 基本構成要素の接続順序

2.6　基本構成要素の接続順序の検討

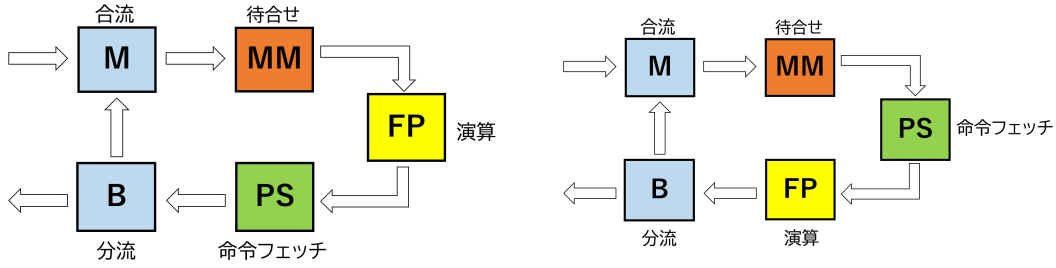
前述したとおり，図 2.6 は基本構成要素の接続順序候補を示している．この 6 通りから有望と考えられる接続順序を検討するために MM に着目する．MM は対応したオペランド対を検出し，オペランド対のデータ同士を結合する構成要素である．すべてのパケットはマッチングをするか否かによらず，必ず MM を通らなければ演算を行うことができない．つまり，MM はこの環状パイプラインの始まりにあたる．そこで図 2.6(a),(c),(e) を例に MM が，基本構成要素の 1 番目，2 番目，3 番目にある時でオペランド対を検出

し演算を行うまでの行程を以下に示す．なお，以下に示すのはオペランド対の片方だけの行程とする．

- (a):M ⇒ MM ⇒ FP
- (c):M ⇒ FP ⇒ MM ⇒ PS ⇒ B ⇒ M ⇒ FP
- (e):M ⇒ PS ⇒ FP ⇒ MM ⇒ B ⇒ M ⇒ PS ⇒ FP

(a) の回路ではパケットが合流した後，すぐに待合せを行い，マッチング後にそのまま演算を行うことができる，しかし (c)(e) の回路では，パケットが合流した後に一度 FP を通り抜けて待合せを行い，マッチング後にパイプラインを 1 周して，再び FP にて演算を行う必要がある．つまり，MM が基本構成要素の 2 番目，3 番目にある時はマッチングしてから演算を行うまでの間にパイプラインを無駄に周回しなくてはならない．また，MM は前述したようにオペランド対を検出して 2 つのパケットを 1 つに結合して出力する構成要素であるため，構成要素 MM が基本構成要素の 2 番目，3 番目に行くに連れて対となるパケットを結合して出力するのが遅くなってしますため，パイプライン内を周回するパケットの流量が多くなってしまう．以上のことから MM は基本構成要素の 1 番目でない場合は性能が低下してしまうため，MM は基本構成要素の 1 番目でなくてはならない．

基本構成要素の 1 番目を MM で固定した場合考えられる組み合わせは図 2.7 に示した 2 通りである．本研究では従来のパイプライン構成である演算後に PS で更新を行う構成を PS 後続型 DDP 構成，本研究で提案する演算前に PS で更新する構成を PS 先行型 DDP 構成と呼ぶ．



(a) PS 先行型 DDP 構成　　(b) PS 後続型 DDP 構成

図 2.7: 2 通りの DDP 構成

2.7　定数演算

図 2.8 に定数演算を行う際の簡易的な動作について示す．

定数演算は入力パケットと CST に格納された定数と演算を行う．それゆえ，定数演算では対となるオペランドと待合せを行う必要がない．CST の各番地には定数演算で使用する定数が保存されている．定数演算を行う際はパケットが保持している dest を用いて CST にアクセスし，定数を取り出す．

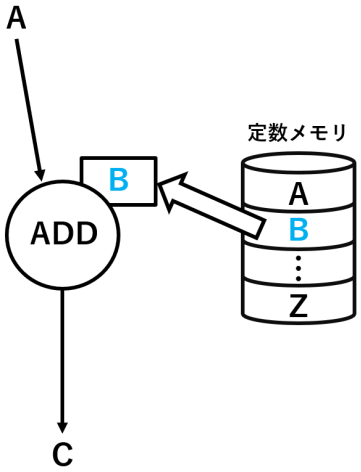
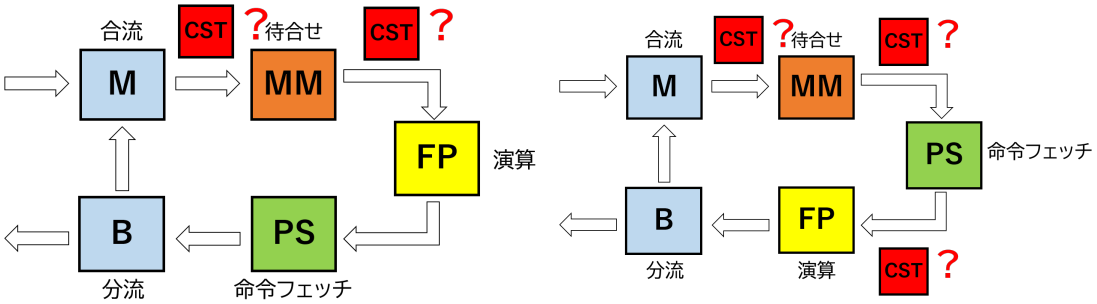


図 2.8: 定数演算

前述したように，定数演算は CST にアクセスし，定数を取り出さないと演算を行うことができない．それゆえ，FP よりも前段に定数読み出しを行う必要がある．よって考え

られる接続順序として図 2.9 の 5 通りが考えられる．



(a) PS 後続型 DDP 構成　　(b) PS 先行型 DDP 構成

図 2.9: 定数読み出の接続候補

2.8　パケット複製機能

図 2.10 に示すように，DDP では同一の演算結果を複製するパケット複製 COPY が備わっている．これにより，同一のデータを保持したパケットをそれぞれ別の宛先へ送ることが可能となる．

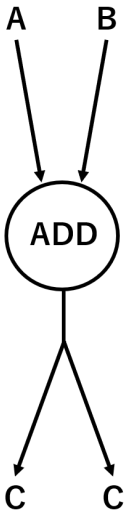
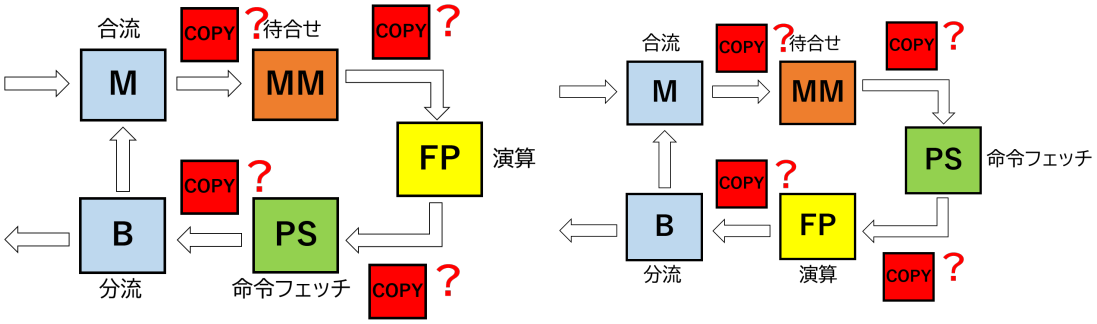


図 2.10: パケット複製 COPY

パケット複製はパケットがパイプラインを周回している最中であればどこでも複製が

可能である．よって，図 4.7 に示す 8 通りが考えられる．



(a) PS 後続型 DDP 構成　　(b) PS 先行型 DDP 構成

図 2.11: パケット複製の接続候補

2.9　結言

本章では，DDP のプログラム上での実行制御，構成要素について説明し，基本構成要素，CST，COPY の接続順序の候補について述べた．それぞれ，基本構成要素は 2 通り，定数読み出しは 5 通り，パケット複製は 8 通りの接続順序が考えられた．また，この接続順序によってそれぞれパケット構成，必要なメモリ量，構成要素の回路が違ってくる．そのためこの組み合わせの中から，最も有望である組み合わせを見つける必要がある．

次章では，前述した接続候補におけるパケット構成，必要なメモリ量，構成要素の回路について述べより有望と考えられるパイプライン構成である PS 先行型 DDP 構成について説明する．

第 3 章

構成要素の接続候補とその検討

3.1 緒言

本章では，前章で述べた，基本構成要素，CST，COPY の接続順序候補から，より有望であると考えられる接続順序を検討した．また，その際，構成要素の回路，パケット構成，メモリ容量等について明らかにし，そのを比較することにより，有望な接続順序と DDP 構成について検討している．

3.2 PS 後続型 DDP 構成とパケットフォーマット

第 2 章で述べたように，DDP は演算ノードを 1 つのパケットとし，構成要素間でパケットの授受を行っている．従来の DDP 構成におけるパケットフォーマットの一例として，パイプライン入力時のパケットフォーマット例を図 3.2 に示す．PS 後続型 DDP 構成は基本構成要素に加え，パケットの入出力を調停する M と B，CST，COPY を加えた七つの構成要素にデータメモリへアクセスする MA を加えた八つの構成要素で構成されている．MA はデータメモリへの書き込みのみを行っている構成要素であるため本章では省略する．図 3.1 に MA を除いた 7 つの構成要素を環状に接続した従来の DDP 構成について示す．本章ではこの接続順序について検討する．

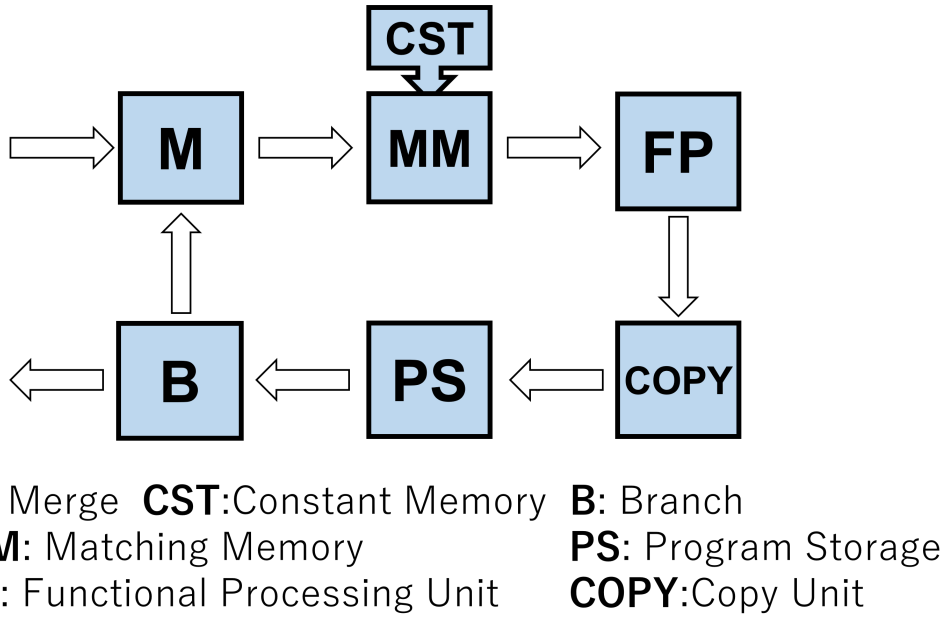


図 3.1: 従来 DDP 構成

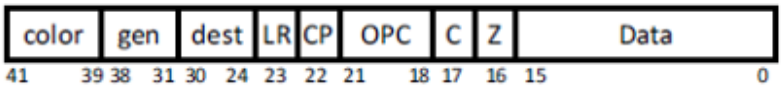


図 3.2: PS 後続型 DDP 構成のパケットフォーマット

3.3　基本構成要素の検討

2.6 で述べたように，基本構成要素は 2 通りの接続順序が考えられる．本章ではこのこの 2 通りの基本構成要素について構成要素の回路，パケット構成，メモリ量について明らかにし比較を行っている．

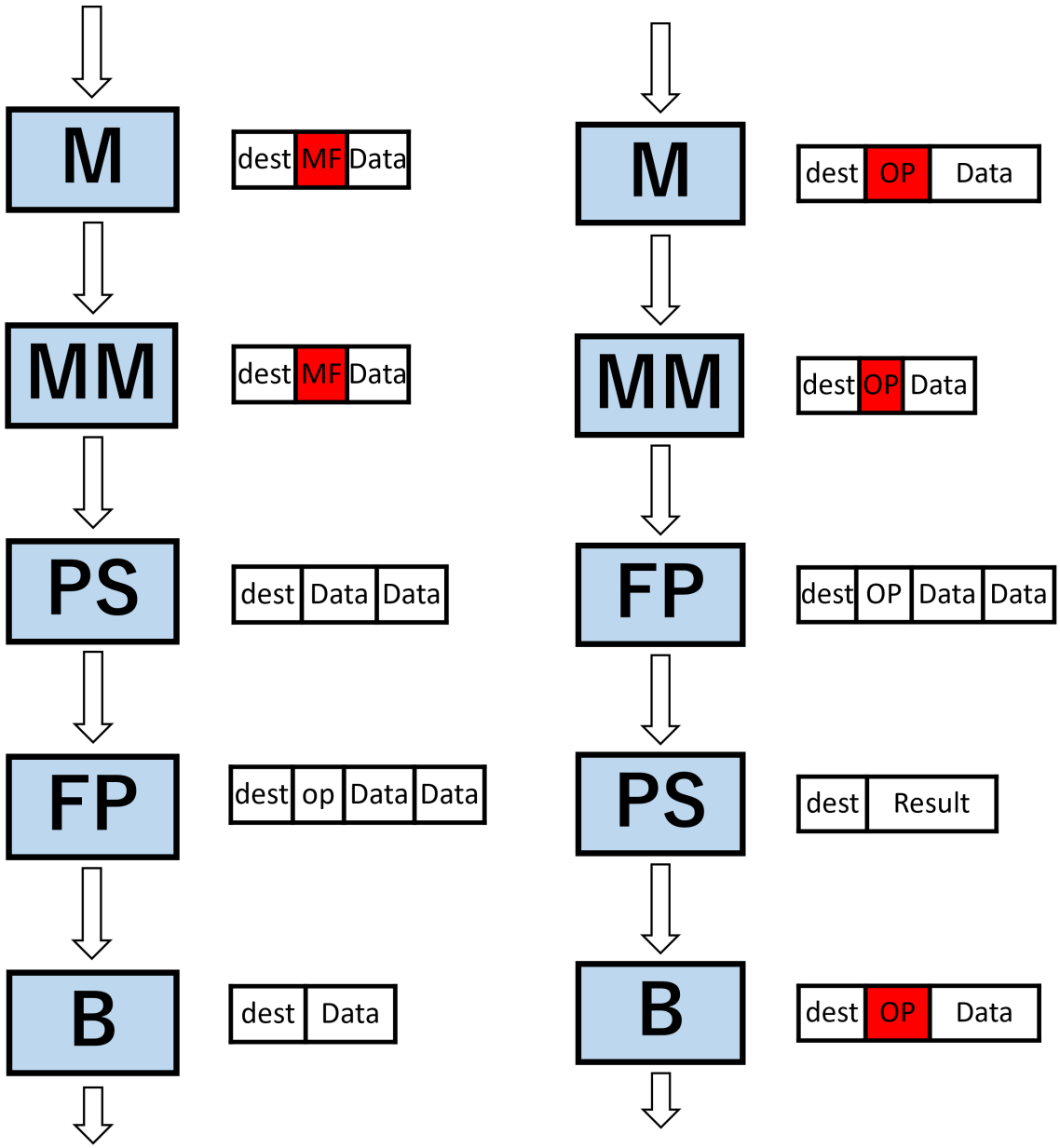
3.3.1　2 通りの接続順序によるパケット構成

PS 先行型 DDP 構成における各構成要素での静的なパケット構成について図 3.3(a) に示す．また，今回は基本構成要素について焦点を当てているため，CST と COPY については省略するものとする．図 3.3 では各構成要素に入力されるパケットの情報を構成

要素の右に示している．また，各構成要素に入力されるパケットのうち，次の構成要素で使われることのない情報を赤色で明記している．

PS 先行型 DDP 構成ではパケットのヘッダを更新後に演算を行うため，パケット入力時に OP を保持する必要がない．それゆえ，周回するパケットは OP の情報分の回路資源を削減でき，ひいては消費電力の削減に繋がる．

これに対して，PS 後続型 DDP 構成では，図 3.3(b) に示すように，演算後にパケットのヘッダを更新するためパケット入力時には OP の半分の情報を保持した状態でパケットが入力され，パイプライン内を周回する必要がある．そのため，M，MM，B で不要な情報を持ったままパケットの授受が行われ，余分な回路資源を使用することになる．



(a) PS 先行型 DDP 構成の入力パケット

(b) PS 後続型 DDP 構成の入力パケット

図 3.3: 各 DDP 構成での入力パケット

3.3.2　PS 先行型 DDP 構成と PS 後続型 DDP 構成の比較

3.3.1 で述べたように，PS 後続型 DDP 構成では 3 つの構成要素に不要な情報を保持したパケットが入力されている．それに対して，PS 先行型 DDP 構成では 2 つの構成要素でのみ不要な情報を保持したパケットが入力される．

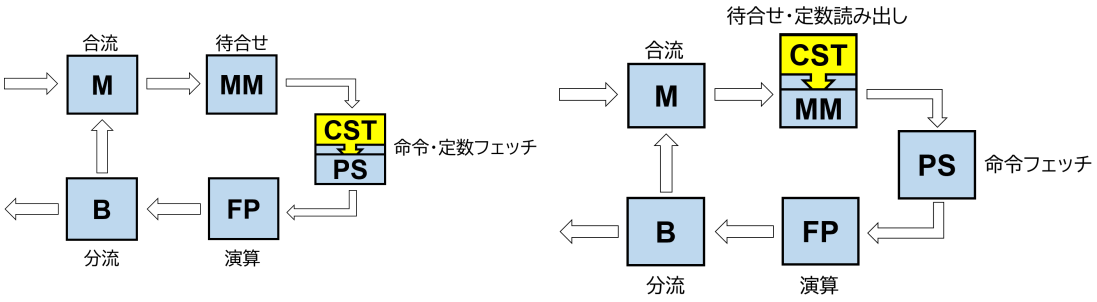
PS 後続型 DDP 構成では，入力時から不要な情報である OP を保持しておかなければならない．また，二項演算の命令コード OP は半分に分割され，各オペランドパケット内に保持されている．この半分に分割された OP はマッチング後にオペランド対がビット結合され 1 つの OP となる．これに対し，PS 先行型 DDP 構成では PS で OP を読み出した直後に演算が行われるため即座に OP の情報が使用され破棄される．しかし，PS 後続型 DDP 構成では半分の OP をフラグとし，待ち合わせを行うかどうかの判断を行っている．そのため，入力時から OP を保持する必要がある．これに対し，PS 先行型 DDP 構成では待ち合わせを行うか否かを判断するフラグである MF(Matching Flag) を新たに追加する必要がある．以上のことから PS 先行型 DDP 構成はパケット長を削減することができるるため，この基本構成要素の接続順序は PS 先行型 DDP 構成が有望であると考えられる．よって以降は PS 先行型 DDP 構成を基準に接続順序について検討する．

3.4　CST の接続順序の検討

CST は定数メモリに dest をアドレスとし，番地に保存されている定数を読み出す構成要素である．また，読み出された定数は，定数演算を行う際に必要となるため，CST は FP よりも前段に接続する必要がある．よって，CST は MM の前後，または PS 先行型における，PS の前後に接続するのが良いと考えられる．それゆえ，CST を含むパイプライン構成は複数通り考えられる．従来の DDP 構成では MM の前段に CST が接続されていた．しかし，定数演算を行う際の定数は MM では必要のない情報となっている．そのため，MM の前段に接続すると不必要な情報を保持したパケットが構成要素間で授受されることとなり，回路規模増加の原因となる．そのため読み出された定数はなるべく少ない構成要素間で授受され，素早く使用され破棄されることが望ましい．

CST は定数メモリへアクセスし，定数を読み出し，結合するだけの構成要素となっている．つまり，必要なのは定数メモリのみであり他の構成要素内で定数メモリの読み出

しを行うことができれば不要な構成要素であるといえる．よって考えられる接続候補は図 3.4 で示すように 2 通りに絞られる．どちらの構成要素に定数メモリを置くかで分けられるため，それぞれ PS 内蔵型と MM 内蔵型と称する．



(a) PS 内蔵型　　(b) MM 内蔵型

図 3.4: CST の結合候補

3.4.1　PS 内蔵型

PS 内蔵型では定数演算を行うための定数を PS メモリ内に保存し,dest,OP 等のヘッダ更新と同時に定数を取り出す方法である．これにより，新たに定数メモリを設ける必要がなくなる．しかし，PS メモリに内蔵することによりメモリ量が定数ビット分増加してしまう増加してしまう．また，図 3.5 に示すように，PS メモリの各番地には dest と OP が保存されているため，定数演算以外の命令時にもメモリにアクセスしてパケットのヘッダを更新する必要がある．PS メモリは可変長ではないためすべての番地において，定数ビット分の領域を確保しなくてはならない．そのため，定数演算以外の演算を行う際はメモリから読み出す値のうち，定数のために確保しているビット分のフィールドが無駄になってしまう．また，定数演算においては，待合せを行う必要がないが，パケットフォーマットは可変長ではないため，定数演算時にはを定数を結合するデータ分のフィールドを余分に確保し MM からパケットを出力する必要がある．そのため，PS 内蔵型は不要なフィールドやメモリの確保が必要となる．

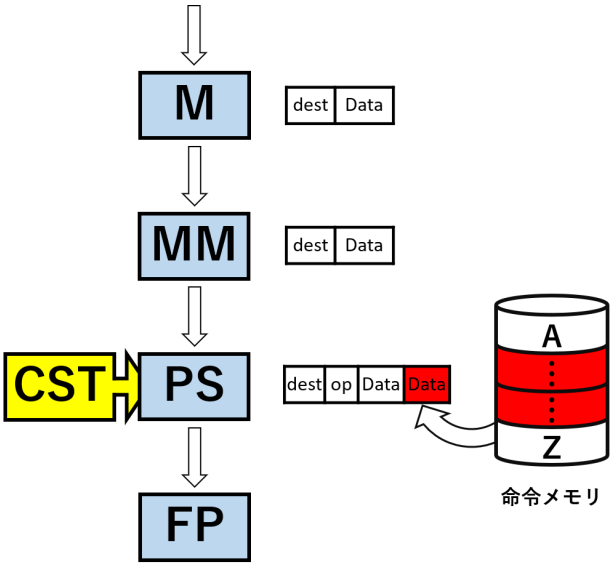


図 3.5: PS 内蔵型

3.4.2　MM 内蔵型

MM 内蔵型では定数メモリを MM 内に確保し，オペランド対の検出，定数演算かどうかの判定を行う際に同時に定数メモリにアクセスして定数を取り出す方法である．これにより，MM 内で二項演算ならばオペランド対を検出し出力され，定数演算ならば定数を結合し出力される．図 3.6 に示すように，MM 内蔵型では MM に定数メモリを内蔵しているため，メモリ構成やメモリの容量に変更はない．また，MM のパケット出力時に二項演算なら 2 つのオペランド対，定数演算ならば定数メモリから読み出した定数を結合し出力されるため，余分なフィールドの確保も必要としない．

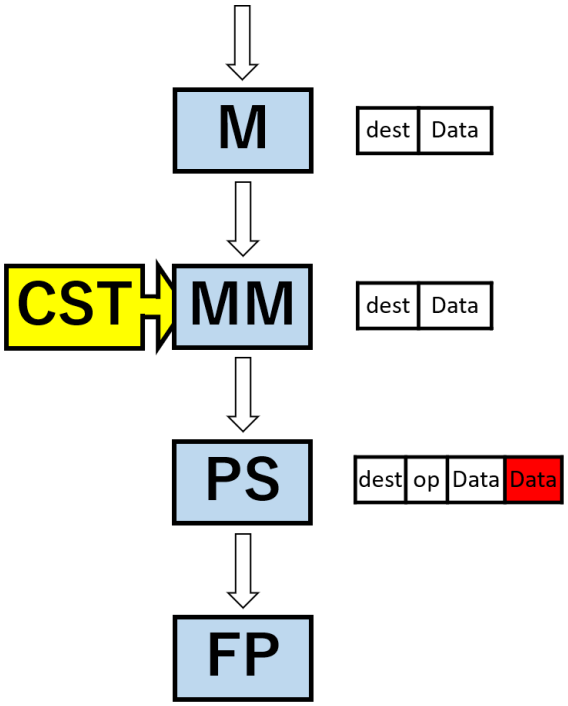


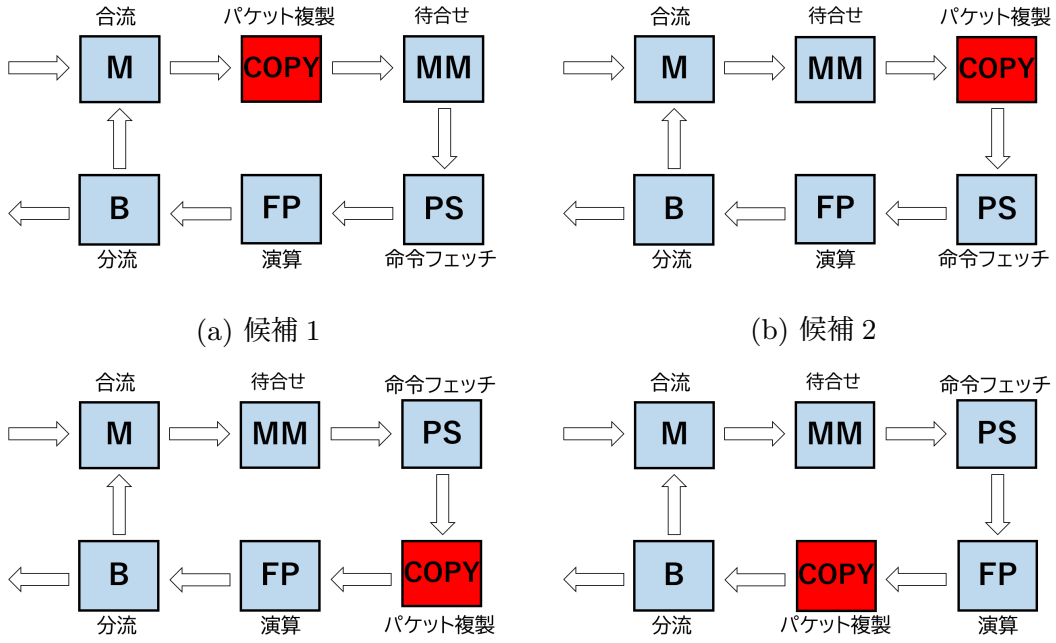
図 3.6: MM 内蔵型

3.4.3　PS メモリ内蔵型と MM 内蔵型の比較

3.4.1 および 3.4.2 で述べたことから，PS メモリ内蔵型では PS メモリの必要量の増加や，不要なフィールドの追加が生じる．一方，MM 内蔵型は MM に定数メモリを置くことによる，メモリ量の増加等のデメリットがない．よって CST は MM に内蔵することが望ましい．

3.5　COPY の接続順序の検討

COPY は同一の演算結果を持ったパケットを複製する構成要素である．パケットをを複製するか否かはパケットが保持している CPY フラグから判断する．前述したように PS 先行型 DDP 構成において COPY は 4 通りの接続候補が考えられる．



(a) 候補 1　(b) 候補 2

(c) 候補 3　(d) 候補 4

図 3.7: パケット複製 COPY の接続順序候補

COPY ではパケットの複製を行うため，複製を行うたびにパイプライン内を周回するパケット流量は増加する．そのため，また，MM ではオペランド対を検出してビット結合を行うため，マッチングするとパケット流量が減少する．この点から，MM の前段に COPY を接続するとパケット流量が増加してしまうことが分かる．よって，COPY は MM の後段に接続することがパケットの流量の観点から見ると望ましい．

回路資源の点から見ると，パケットの複製を行うか否かを判断するフラグをすぐに使用し，破棄できるため PS の後段に COPY を接続することが望ましい．よって，二つの観点から MM ⇒ PS ⇒ COPY の接続順序が最も有望であるといえる．また，パケット複製は同一の演算結果を持ったパケットを複製する際に使用することが多いため，FP よりも後段に接続することが望ましい．以上のことから考えられる組み合わせは図 3.8 に示す接続順序が最も有望であると考えられる．

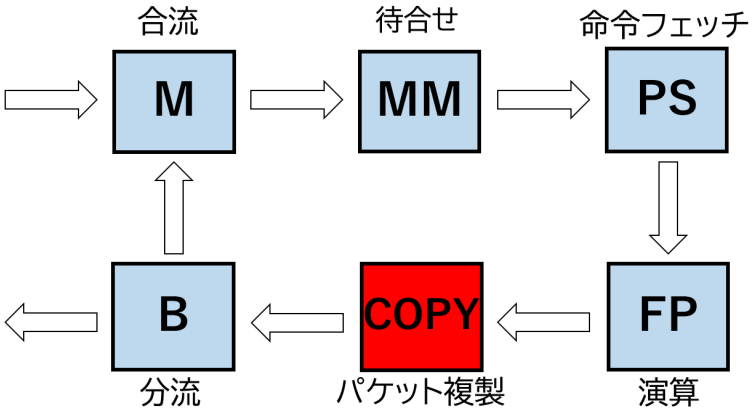


図 3.8: 有望な接続候補

3.6　結言

本章では PS 先行型 DDP 構成における，基本構成要素，定数読み出し，パケット複製の有望な接続順序の検討について構成要素の回路，パケット構成，メモリ容量，パケットの総量を明確にすることにより明らかにした．基本構成要素，定数読み出し，パケット複製の有望な接続順序を組み合わせることにより最終的な PS 先行型 DDP 構成を決定している．

よって次章では，従来の PS 後続型 DDP 構成と本研究で提案する PS 先行型 DDP 構成との比較について説明する．

第 4 章

パケット流量と稼働率

4.1 緒言

本章では DDP の構成要素間で授受するパケット流量について述べる．パイプライン内を周回するパケットの総流量は，プログラム中の演算ノード間のデータ依存関係により増減する．また，構成要素で授受されるパケット流量はパイプライン構成によって変動することが明らかとなっている．本研究では n 入力 m 出力であり，マッチングを必要とする二項演算，定数を対象とするマッチング不要な二項演算，および，演算結果の複製を含む任意の非巡回データ依存関係を有するデータ駆動型プログラムを対象として，環状パイプライン内を流れるパケット数のモデル化を検証している．パケット流量の増減は各構成要素の稼働率の増減に繋がる．ゆえに，稼働率の増加は消費電力の増大に繋がる．よって本章ではパケット流量のモデル化とその手法について明らかにしている．

4.2 DDP のパケット流量

パイプライン内を周回するパケットはプログラム中の演算ノード間のデータ依存関係応じて増減する．図 4.3 に示すように，二項演算時には 2 つのパケットが入力され，マッチング後にビット結合し 1 つの演算結果として出力される．このように，プログラム中の演算ノードによりパケットの流量は変化する．同様に定数演算，パケット複製等が挙げられる．二項演算時は MM の前後でパケット数が 1 減少する，また，パケット複製時は COPY の前後でパケットが 1 増加することとなる．定数演算時は，パケットと定数

で演算が行われるため，パケット流量の増減は起こらない．その他にも，パケットの合流部 M の後では，合流したパケット数分が DDP での演算対称に加わるためその都度パケット流量が増加する．一方，パケット分流部 B では，環状パイプラインを周回してきたパケットが再度演算対象となるか，あるいは，外部へ出力されるかを判断するため，出力したパケット分パケット流量が減少することとなる．

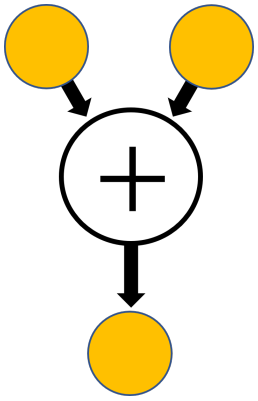


図 4.1: 二項演算時のパケット

4.3　パケット流量のモデル化

前述したとおり本研究では n 入力 m 出力を想定した DDP を設計している．この際，プログラム上ではパイプライン構成によってパケットの総量に変化は見られない．つまり，n 入力 m 出力の演算を行うプログラムでは，パイプライン構成を変えてもプログラム実行に伴うパケット総量は変わらない．しかし，パイプライン内のパケット流量に着目すると，各構成要素間で授受されるパケット流量に違いが見られる．

4.3.1　二項演算時のパケット流量

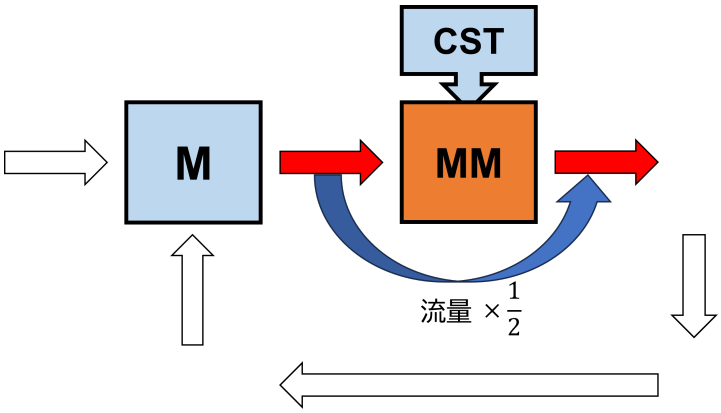


図 4.2: 二項演算における MM のパケット流量

前述したように，構成要素間で授受されるパケット量には違いがある．つまり構成要素の接続順序を変更することにより，パイプライン構成が変わるとパイプライン内のパケット流量が増減する．

図 4.2 に示すように，二項演算を行う際のパイプライン内のパケット流量に着目していく．まず初めに，入力パケット数を n とした際の M ⇒ MM 間で授受されるパケット量について考えていく．この際計算しやすいように出力されるパケットを 1 とする．つまり入力されたパケットは出力が 1 になるまでパイプライン内を周回し，二項演算が行われる．MM では 2 つのパケットがマッチングしビット結合され，1 つのパケットとなり出力される．仮に，入力された全てのパケットがマッチングすると考えると出力されるパケット数は，入力されたパケット数 n に対して，ちょうど半分の $\frac{n}{2}$ となる．また，その全てのパケットが周回し，再び MM に入力されるため，次の入力パケット数は $\frac{n}{2}$ となる．このように MM へはパケットが残り 2 つになるまで入力され続ける．その後，マッチングしたパケットは外部へ出力される．この際，M ⇒ MM 間のパケット流量は $n + \frac{n}{2} + \frac{n}{4} + \cdots + 2$ となる．M ⇒ MM 間のパケット流量の一般式はマッチング回数から算出することが可能である．マッチング回数とはつまり，$\log_2 n$ 同義である．よって

求める式は

$$\sum_{k=1}^{\log_2 n} \frac{n}{2^{k-1}} \tag{4.1}$$

となる．式 4.1 より入力パケット数 n が 2 で割れる回数 S は

$$S = n - 1 \tag{4.2}$$

となる．式 4.2 で算出されるのはマッチング回数である．この式よりパケット流量は マッチング回数× 2 であるため，M ⇒ MM 間で授受されるパケット量は $2(n-1)$ で表すことができる．また，図 4.2 で示したように，MM から出力されるパケット流量は入力の半分の量となるので MM ⇒ B 間で授受されるパケット量は $n-1$ と表すことができる．また，出力されるパケット以外はパイプライン内を周回するため，再び M へ入力される．そのため，$n-2$ と表すことができる．本研究では n 入力 m 出力の DDP を想定しているため出力パケット数を m とすると二項演算時のそれぞれの構成要素間で授受されるパケット量は図 4.3 に示す通りとなる．ただし，n，m，はそれぞれ入力パケット数，出力パケット数である．

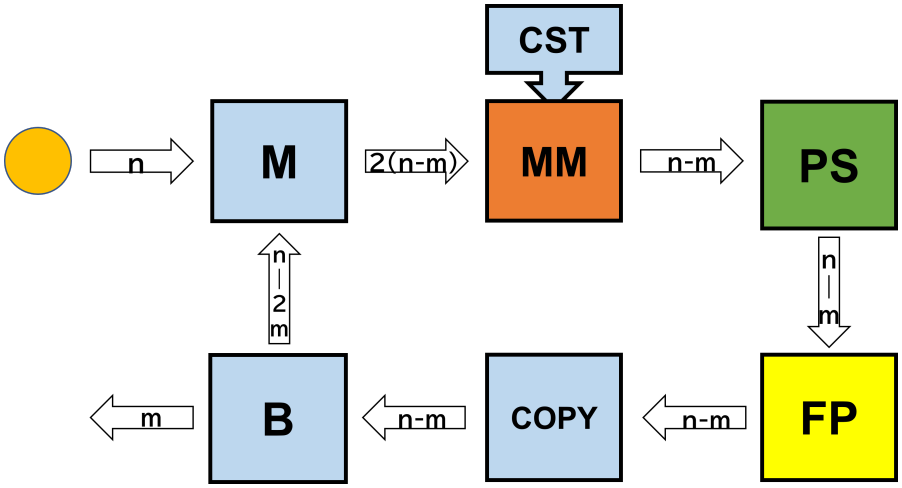


図 4.3: 二項演算時のパケット

4.3.2　定数演算時のパケット流量

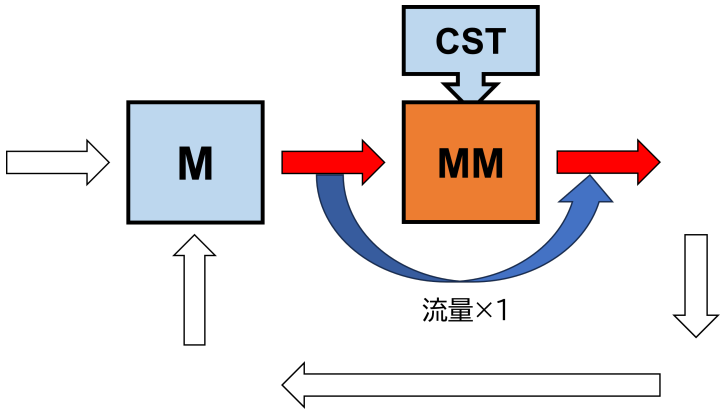


図 4.4: 定数演算における MM のパケット流量

図 4.4 に示すように，定数演算では定数メモリから読み出した定数と演算を行うため，マッチングを必要としない．そのため定数演算時に MM に入力されるパケット数と出力するパケット数は同数である．よって定数演算時のそれぞれの構成要素間で授受されるパケット量は図 4.5 に示す通りとなる．なお u は定数演算数を表している．

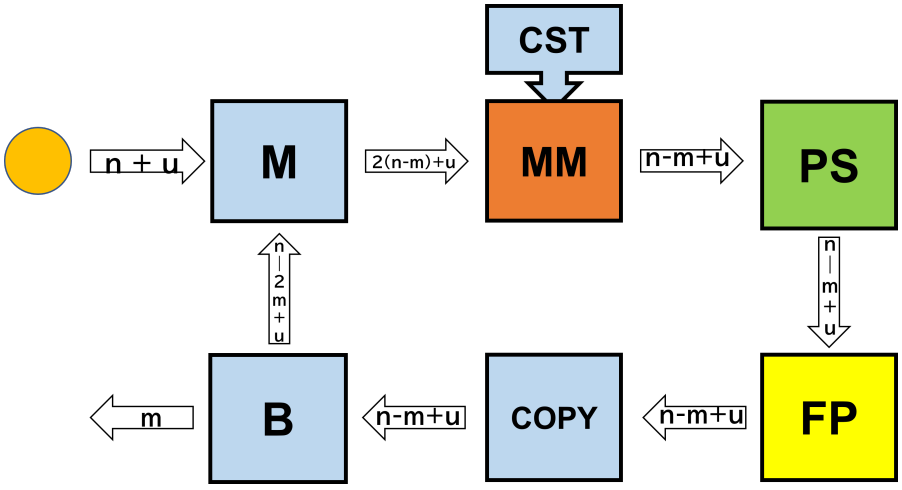


図 4.5: 定数演算時のパケット

4.3.3　パケット複製時のパケット流量

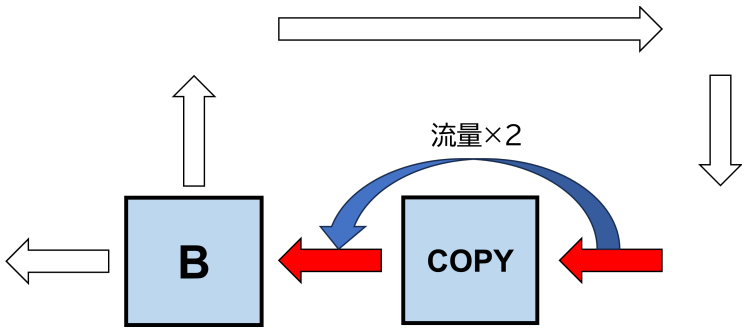


図 4.6: パケット複製における COPY のパケット流量

図 4.6 に示すように，パケット複製では同一演算結果を持ったパケットを複製するため，COPY では入力パケット数 n の 2 倍のパケット数が出力される．よってパケット複製時のそれぞれの構成要素間で授受されるパケット量は図 4.7 に示す通りとなる．なお c はパケット複製回数を表している．

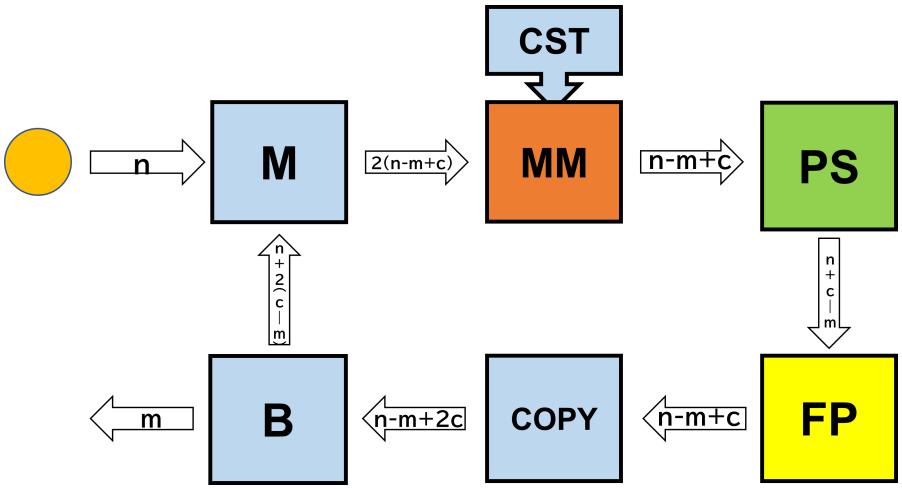


図 4.7: パケット複製時のパケット

4.3.4　パケット流量モデル

4.3.1，4.3.2，4.3.3，よりパケット流量の簡易モデルは図 4.8 のように表すことができる．

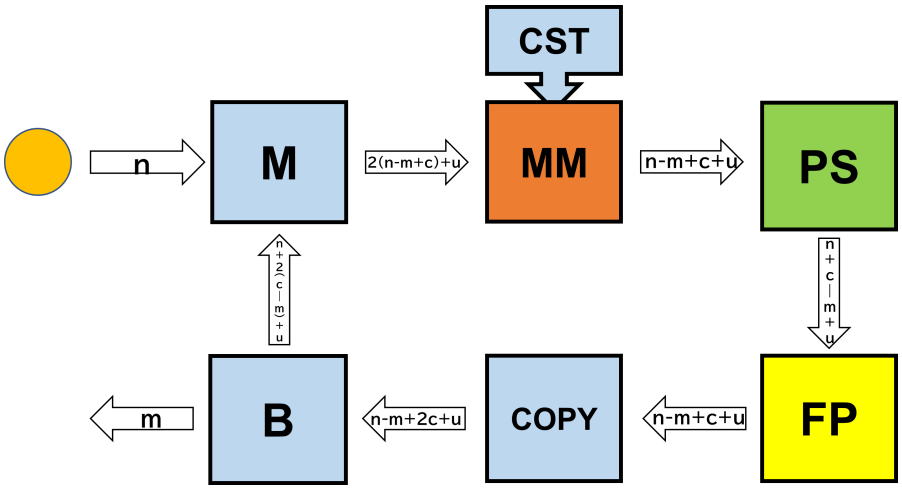


図 4.8: パケット複製時のパケット

構成要素間で授受されるパケット流量を明らかにすることにより，パケットを授受される際に使用される回路資源の頻度 (稼働率) を明確にすることが可能となる．稼働率を算出することにより回路資源の使用頻度から消費電力量を算出することができる．そして，PS 先行型 DDP 構成と PS 後続型 DDP 構成の消費電力量を比較することにより，消費電力の評価と比較を行うことができる．

4.4　結言

本章では PS 先行型 DDP 構成における，構成要素間で授受されるパケット流量についての簡易モデルを明らかにした．構成要素間のパケット流量が定量化できることにより，その間を通るパケットの頻度から構成要素の稼働率を算出することが可能となる．つまり，頻繁にパケットが入力される構成要素は多くの演算を行うため稼働率が高くなり，ひいては消費電力の増加に起因する．また，前述したとおり回路規模削減率と稼働率をかけ合わせることにより DDP の総消費電力量が概算可能となる．

よって次章では，PS 後続型 DDP 構成と PS 先行型 DDP 構成との評価・比較について説明する．

第 5 章

評価

5.1 緒言

本章では，PS 先行型 DDP 構成と PS 後続型 DDP 構成の設計を AMD 社製 zynq7000 をターゲット FPGA として設計し評価を行った．回路規模の評価は，zynq7000 に用意されている回路資源である，LUT，FF，BRAM の使用数で評価する．なお，BRAM は 1 個あたり 36Kb 容量の RAM である．また，消費電力は 回路規模削減率×稼働率 から総消費電力量を概算し評価している．

5.2 PS 先行型 DDP 構成の仕様

DDP では構成要素の接続順序によってパケット長，メモリ量が異なる．よって本研究で提案する PS 先行型 DDP 構成の設計仕様について，共通パケット仕様，非共通パケット仕様，共通命令セット，メモリ量を下図に示す．なお，非共通パケット仕様については，パイプライン構成によって，ビット長が異なったり，要・不要なフラグがあるためである．

信号名	bit
color	3
gen	8
dest	7
flags	5
Data	16

表 5.1: 共通パケットの設計仕様

信号名	bit
OP	4/6/7
MF	0/1
LR2	0/2

表 5.2: 非共通パケットの設計仕様

命令種別	ニーモニック	意味
算術論理演算	ADD	加算
	ADDC	
	SUB	減算
	SUBC	
	AND	論理積
	OR	論理和
	XOR	排他的論理和
	MUL	乗算
	SHL	論理左シフト
	SHR	論理右シフト
	ROL	左ローテート
	ROR	右ローテート
メモリアクセス	LDM	ロード命令
	STM	ストア命令

(a) 算術論理演算・メモリアクセス

命令種別	ニーモニック	意味
分岐命令	BZ	ゼロ分岐
	BNZ	非ゼロ分岐
	BC	桁あふれ分岐
	BNC	非桁あふれ分岐
	BZL	ゼロ分岐
	BNZL	非ゼロ分岐

(b) 分岐命令

命令種別	ニーモニック	意味
その他	ABSORB	パケット消去
	CHGCOL	color変更
	ADDGEN	gen加算
	DECGEN	Gen減算

(c) その他の命令

メモリ種別	深さ	PS先行型DDP構成	PS後続型DDP構成
定数メモリ	64 words	16bit	20bit
マッチングメモリ	64 words	20bit	24bit
データメモリ	1024 words	16bit	16bit
PSメモリ	128 words	17bit	13bit
PS2メモリ	64 words	1bit	-

表 5.4: メモリの設計仕様

5.3　回路規模の評価

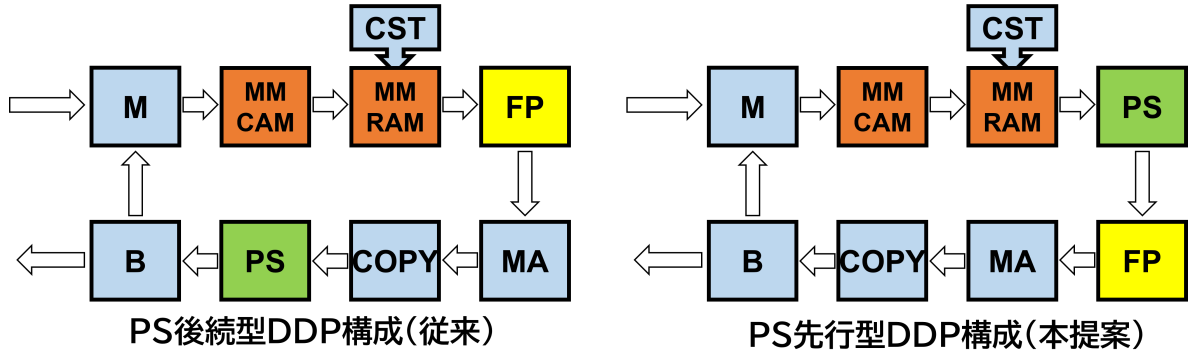


表 5.5: 評価対象となる DDP

従来の DDP 構成と本論文で提案する PS 先行型 DDP 構成を図 5.5 に示す．評価対象となる DDP は基本構成要素，定数読み出し，パケット複製，データメモリアクセス，パケット入出力管理の計 9 個の構成要素を環状に接続したパイプライン構成を対象としている．本研究で設計した回路は AMD 社の zynq7000 FPGA 上への実装を想定して設計されている．zynq7000 では有限な回路資源として，LUT，FF，BRAM が用意されている．また，それぞれの回路資源の使用可能個数を図 5.6 に示す．また，BRAM は 1 ブロックあたり 32Kb の容量のメモリである．また，本研究では各 DDP 構成におけるマッチングメモリを BRAM を使用して実装した場合とレジスタを使用して実装した場合の 2 パターン設計し評価している．

	回路資源数
LUT	17600個
FF	35200個
BRAM	80個

表 5.6: 使用可能な回路資源数

各 DDP 構成の回路を実際に設計し，各パイプライン構成順序による設計回路規模を LUT，FF，メモリ量の使用数から比較し，図 5.7 に示す．

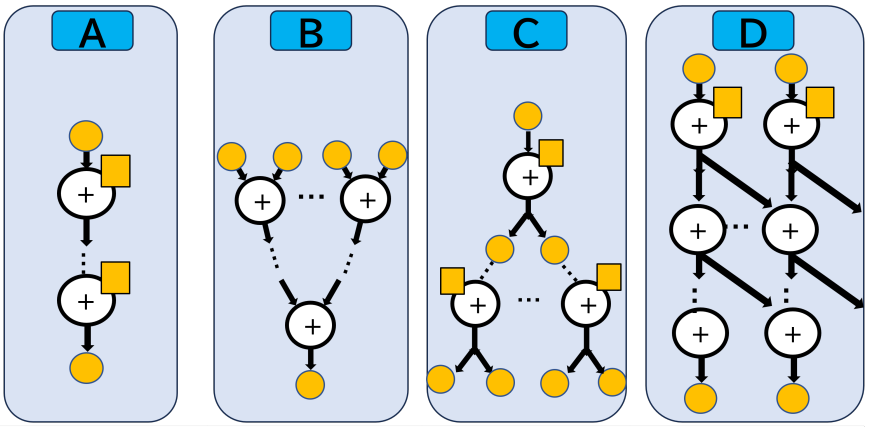
	PS先行型DDP(本提案)		PS後続型DDP(従来型)	
	RAM	REG	RAM	REG
LUT	4164個	1906個	5474個	2102個
FF	475個	1680個	590個	1681個
BRAM	8.5個 (272Kb)	2.5個 (90Kb)	6.5個 (236Kb)	2個 (64Kb)

表 5.7: 各 DDP 構成の回路資源の使用数

結果，レジスタ型では LUT 数は 2102 個から 1906 個へ 10%の削減，FF は 1681 個から 1680 個へ 0.06%の削減に成功した．しかし，メモリ量に関しては BRAM が 0.5 個増加するといった結果となった．また，RAM 型では，LUT 数は 5474 個から 4164 個へ 31%の削減，FF は 590 個から 475 個へ 24%の削減に成功した．しかし，メモリ量に関しては BRAM が 2 個増加するといった結果となった．以上のことからレジスタ型でも RAM 型でも提案する DDP 構成のほうが回路規模を削減できる．しかし，RAM 型では使用する回路資源数が大幅に増加してしまっている．これは，マッチングメモリを BRAM で実装するにあたり，大幅に回路資源を使用してしまっていることが原因であると考えられる．そのため，RAM 型の DDP 構成は見直す必要があると考えられる．

5.4　消費電力の評価

本研究では DDP の回路資源使用数から算出した回路規模削減率と構成要素間で授受されるパケット流量から算出される稼働率を基に 回路規模削減率×稼働率 から総消費電力量を概算し，消費電力の比較・評価を行っている．なお概算する際のプログラムはそれぞれ極端な場合を想定してプログラムを設定している．評価に使用したプログラムを図 5.9 に示す．図に示すように，それぞれ A〜D までの 4 パターンのプログラムを用意している．これらのプログラムを 4.3.4 で明らかにした簡易モデルを使用し総消費電力量の概算を行っている．



	A	B	C	D
入力数n	1	64	1	32
出力数m	1	1	32	32
定数演算数u	64	0	63	32
パケット複製回数c	0	0	63	64

表 5.8: 実行プログラム

(A) 1 入力 1 出力のプログラム．このプログラムでは，定数演算を 64 回繰り返し，出力するプログラムとなっている．

(B) 64 入力 1 出力のプログラム．このプログラムでは，入力された全てのパケットが出力が 1 となるまで二項演算を繰り返すプログラムとなっている．

(C) 1 入力 32 出力のプログラム．このプログラムでは，1 入力に対して，定数演算を行った後演算結果を複製，を繰り返している．この操作を 63 回行うことにより 32 個のパケットを出力するプログラムとなっている．

(D) 32 入力 32 出力のプログラム．このプログラムでは，入力されたパケットと定数演算が行われその後パケットが複製されそれぞれ別のノードへ送られる．これを繰り返すことにより，32 個のパケットを出力するプログラムとなっている．

このように 4 つの極端なパターンから総消費電力量を概算している．また，定数演算数が多い時や二項演算が多い際の総消費電力量の差異についても明らかにし，図 5.9 に示す．

	正規化総消費電力量			
	PS先行型DDP		PS後続型DDP	
	RAM	REG	RAM	REG
プログラムA	378K	231K	384K	243K
プログラムB	660K	374K	668K	399K
プログラムC	737K	443K	745K	471K
プログラムD	890K	522K	1019K	583K

表 5.9: 各プログラムごとの総消費電力量

結果，本研究で提案する PS 先行型 DDP 構成は PS 後続型 DDP 構成と比較してどのようなプログラムでも総消費電力量が減少することが分かった．特に，プログラム C とプログラム D は総消費電力量の削減率が大きかったことから，パケット複製における消費電力量の削減量が最も多きいことが分かる．しかし，RAM 型とレジスタ型では回路規模が大きく異なることから全体的に消費電力量が増大してしまうといった結果になった．

5.5　結言

本章では従来の DDP 構成である PS 後続型 DDP 構成と本研究で提案する PS 先行型 DDP 構成との比較・評価を行った結果について述べた．また，それぞれの DDP 構成でマッチングメモリを実装する際に，BRAM を使用して実装する RAM 型とレジスタを使用して実装するレジスタ型の計 4 種を評価対象とした．結果として，RAM 型では LUT 数は 31%の削減，FF は 24%の削減に成功した．レジスタ型では LUT 数は 10%の削減，FF は 0.06%の削減に成功した．しかし，どちらも BRAM 数は増加してしまった．以上のことから，削減率に差はあるものの RAM 型とレジスタ型それぞれで回路資源数を削減することに成功した．また，BRAM 数の増加に関しては，パイプライン構成を変更したことによりメモリ構成が変わり保存するべき情報が増加したためであると考えられる．評価結果から PS 先行型 DDP 構成は従来の DDP 構成より有望であると言える．

第 6 章

結論

6.1　本論文のまとめ

近年，IoT(Internet of Things) 技術の普及に伴い，「医療」，「コンシューマー」，「産業用途」及び「自動車・宇宙航空」業界においての IoT 機器は今後，急激に成長することが予想されている，これらの業界では，IoT デバイスによるセンサでのリアルタイム情報の取得やデータの高速処理，デバイスの長時間稼働が期待されている．そのため，IoT デバイスには，高性能かつ省電力が求められている．STP 技術を用いて多重並列処理が可能で，局所的な動作を行える DDP を用いた回路設計を行うことによりこの 2 つの条件を満たすことが可能である．

データ駆動型プロセッサはプログラム実行に必要な構成要素を環状に接続したパイプラインで構成されている．しかし，その構成要素の接続順序により，回路規模や性能にも差が生じてしまう．そのため，より有望と考えられる接続順序で環状に接続されたパイプラン構成の検討が必要である．

パイプライン構成の接続順序を検討する際，二項演算に最低限必要となる MM，PS，FP の 3 つの基本的構成要素と定数演算に必要な定数メモリから定数を読み出す CST，パケット複製を行う COPY の 3 つに分けてそれぞれの最適な接続順序について検討した．結果として，基本的構成要素は MM ⇒ PS ⇒ FP の順序が最も有望である．定数演算に関しては，MM に定数メモリを内蔵するのが最も有望である．パケット複製に関しては MM ⇒ PS ⇒ COPY の接続順序が最も有望であることが分かった．

以上のことから PS 先行型 DDP 構成が最も有望な接続順序である．実際に IoT 向け

DDP 仕様を対象として AMD 社の zynq7000 をターゲットに FPGA 回路設計を行い，従来の DDP 構成である PS 後続型 DDP 構成と比較評価を行った．RAM 型では LUT 数は 31%の削減，FF は 24%の削減に成功した．レジスタ型では LUT 数は 10%の削減，FF は 0.06%の削減に成功した．また，パイプライン内に着目し，構成要素間で授受されるパケット流量から稼働率の簡易モデルを算出し，回路削減率と稼働率から総消費電力量を概算した．この際，4 つの極端なプログラムを使用し，いかなる場合でも PS 先行型 DDP 構成は総消費電力量を削減できる．

6.2　今後の課題

今後の課題として評価の精度向上が必要である．回路規模においては遅延素子をしっかりとチューニングする必要がある．また，厳密に回路規模を評価するにあたり，配置・配線の方法や固定方法もしっかりと検討する必要がある．本研究では使用回路資源数を基に回路規模の評価を行っていたが，より精度を向上するために回路面積にも目を向けるべきであると考える．遅延素子のチューニングにより，より厳密で高速な動作が可能となる．また，回路規模縮小は単純に回路資源の使用数を削減するだけでなく，配置・配線を工夫することでも可能である．そのため評価の精度向上から，DDP に適した配置・配線を検討することにより，回路規模はさらに削減できると考えられる．また，回路面積の縮小は，マルチコア化においてたいしょうとなる FPGA のコア数に直接影響を与える．本研究ではシングルコアでの検討を行っているが，より大規模なデータをエッジ端末として処理する際には，マルチコア化は必須となる．その際に，コア数の多いエッジ端末の開発において，回路規模の削減と回路面積の縮小は重要な要件であると考えられる．

次に，性能評価の精度向上が挙げられる，本研究では簡易的な性能評価を行ったが，決して精度が高い評価方法とは言えない．そのためプログラムの実行性能等，より精度の高い性能評価が必要である．最後に総消費電力量に関して，より高精度な見積もり方法

の検討が必要であると考える．一番の理想は実測値の測定であるが，現状非同期回路の実測値の測定は困難である．そこで実測値に近い高精度な総消費電力量の見積もりは省電力回路の研究を行う上でも今後の重大な課題であると考えられる．また，回路規模同様，マルチコア化において性能評価の精度向上は重要な要件の 1 つである．性能評価の精度向上により，今後の改良すべき問題点がより浮き彫りになると考えられる．

謝辞

本研究に際して，様々なご指導をいただきました岩田誠教授に深く感謝申し上げます．お忙しい中，貴重なお時間を割いて頂き，時には遅い時間まで相談に乗って頂いたおかげで本研究を進めることができました．研究だけでなく，就職活動においても，客観的な意見や就職活動に置ける様々なサポートを頂き，勉学だけにとどまらず，進路相談までご協力いただき誠にありがとうございます．また，本研究の副査を引き受けてくださった，福本昌弘 教授並びに，横山和俊 教授に心から感謝申し上げます．

そして，同研究室メンバーとして日頃から暖かいご支援ご協力を頂きました，張震 氏，Valeeprakhon Tamnuwat 氏，古田雄大 氏，植元陸 氏，坂口白磨 氏，松坂拓海 氏，市ノ木一希 氏，伊藤雅俊 氏，岡村健勝 氏，山下拓巳 氏，大綾斗 氏，奥平舜理 氏，片岡拓心 氏，門屋陽丈 氏，山浩正 氏に心より感謝致します．私が卒業した後も研究室に在籍される後輩の皆様におかれましては，もし思い浮かべば私の良いところを真似していただき，基本的には反面教師として，失敗の少ない有意義で充実した学生生活を送っていただけたらと思います．

最後になりましたが，岩田誠教授をはじめ，日頃からご支援いただきました関係者の皆様，日常生活を支えていただいた友達，両親，弟に心より御礼申し上げます．

参考文献

[1] 総務省, “令和 3 年版 情報通信白書 IoT デバイスの急速な普及, ” https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r03/pdf/n0100000_hc.pdf, 2024 年 2 月 2 日 参照.

[2] 総務省, “令和 5 年版 データ集, ” https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r05/html/datashu.html#f00213, 2024 年 2 月 2 日 参照.

[3] 三宮 秀次, 青木 一浩, 宮城 桂, 岩田 誠, 西川 博昭,“超低消費電力化データ駆動ネットワーキングプロセッサ ULP-CUE の試作とその評価, ” 情報処理学会論文誌, コンピューティングシステム Vol.6 No.1 78-86(Jan. 2013), https://ipsj.ixsq.nii.ac.jp/ej/?action=repository_action_common_download&item_id=89939&item_no=1&attribute_id=1&file_no=1

[4] 総務省, “データ主導経済と社会確変, ” https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h29/html/nc133220.html, 2024 年 2 月 2 日 参照

[5] Kei MIYAGI, Shuji SANNOMIYA, Makoto IWATA, Hiroaki NISHIKAWA,“Low-Powered Self-Timed Pipeline with Runtime Fine-Grain Power Supply, Proc. 2012 Int’l Conf. on Parallel and Distributed Processing Techniques and Applications, pp.472-478, Las Vegas, U.S.A., July 2012.

[6] Kei MIYAGI, Shuji SANNOMIYA, Keiichi SAKAI, Makoto IWATA,“Autonomous power-supply control for ultra-low-power self-timed pipeline, ” Proc. 2008 Int’l Conf. Parallel and Distributed Processing Techniques and Applications, pp.704–709, Las Vegas, Nevada, U.S.A., July 2008.

[7] 宮城 桂, 岩田 誠, 三宮 秀次, 西川 博昭,“細粒度パワーゲーティング機構を備えた自己同期型パイプラインとその実装評価, ” 電子情報通信学会

論 文 誌, Vol.2021-ARC-243 No.25, Vol.2021-SLDM-193 No.25, 2021/1/26, https://ipsj.ixsq.nii.ac.jp/ej/index.php?active_action=repository_view_main_item_detail&page_id=13&block_id=8&item_id=209261&item_no=1

[8] 吉川 千里, 三宮 秀次, 岩田 誠, 佐藤 聡, 西川 博昭,“FPGA 向き自己同期型パイプライン回路構成法, ” 情報処理学会研究報告, Vol. J97–A No. 8 pp. 554–564, 2014/8, https://search.ieice.org/bin/pdf_link.php?category=A&lang=J&year=2014&fname=j97-a_8_554&abst=

[9] H. Terada, et al., “DDMP's: Self-Timed Super-Pipelined Data-Driven Processors,” *Proc. of the IEEE*, Vol. 87, No. 2, pp. 282–296, Feb. 1999.

[10] 髙橋龍一, “データ駆動型プロセッサの環状パイプライン構成の比較検討,” KUT 学士学位論文, 2022.

[11] H. Terada, S. Miyata, and M. Iwata, “DDMP's: Self-Timed Super-Pipelined DataDriven Multimedia Processors,” Proceedings of the IEEE, Vol. 87, NO. 2, pp. 282—296, Feb. 1999.

[12] 松阪拓海, “データ駆動型プロセッサの FPGA 向け電力効率改善手法の検討,” KUT 学士学位論文, 2023.