令和 7 年度

修士学位論文

負荷変動耐性を備えた
データ駆動型プロセッサの構成法

A Data-Driven Processor Architecture
With Load Fluctuation Tolerance

山下　拓巳

指導教員　岩田　誠

2026 年 2 月 27 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

負荷変動耐性を備えた
データ駆動型プロセッサの構成法

山下 拓巳

近年，IoT(Internet of Things) 技術の普及に伴い，アプリケーションが高度化・多様化し，高性能かつ省電力なプロセッサが求められている．

この要件を満たす技術として，データ駆動型プロセッサ DDP(Data-Driven Processor) が注目されている．DDP はその動作原理からデータ依存関係にない処理を自然に並列実行可能で，さらにセルフタイム型パイプライン STP(Self-Timed Pipeline) 回路で実装することで省電力な動作が可能である．一方，DDP では周回パイプライン内を巡回するデータパケットの流れを円滑にしないとパイプライン処理性能を充分に発揮できず，最悪の場合，パケット転送が停止するといった問題を内在している．この問題に対して，先行研究では，巡回パケット数の変動を吸収可能なキューバッファ機構 QB(Queue Buffer) を DDP パイプライン内に導入する方式が提案されている．しかし，この研究では，どの程度の容量の QB を導入すれば，どの程度の負荷変動耐性を備えられるのかは，十分には明らかにされていない．

そこで，本研究では，応用プログラムの実行時の負荷変動状況に基づいて QB の適切な容量およびプログラム実行性能を見積ることによって，負荷変動耐性を備えたデータ駆動型プロセッサを構成する方法を提案した．結果，プログラムの特徴をランク毎に捉えることによって，各 QB 容量およびプログラム実行性能について，上界を特定できる見通しが得られた．

キーワード　IoT(Internet of Things)，データ駆動型プロセッサ，セルフタイム型パイプライン，キューバッファ，負荷変動耐性

Abstract

A Data-Driven Processor Architecture With Load Fluctuation Tolerance

Takumi YAMASHITA

In recent years, with the proliferation of IoT (Internet of Things) technology, applications have become increasingly sophisticated and diverse, demanding high-performance power-efficient processors.

To meet these requirements, the Data-Driven Processor (DDP) has garnered significant attention. Due to its operating principle, the DDP can naturally execute data-independent operations in parallel. Furthermore, implementing it with a Self-Timed Pipeline (STP) circuit enables low-power operation. However, the DDP inherently faces a problem: if the flow of data packets circulating within the pipelined stages is not smooth, the pipeline processing cannot fully perform. In the worst case, packet transfer may halt. To address this issue, prior research has proposed introducing a queue buffer (QB) mechanism within the DDP pipeline to absorb fluctuations in circulating packets. However, this research did not sufficiently clarify how much load fluctuation tolerance can be achieved with a QB of a given capacity.

Therefore, this study proposes a method for designing a load-fluctuation-tolerant data-driven processor by estimating the appropriate QB capacity and program execution performance based on the load fluctuation conditions during application program execution. As a result, by characterizing a program structure by its rank, we obtained the prospect of identifying upper bounds for each QB capacity and program execution

performance.

key words IoT(Internet of Things), Data-Driven Processor, Self-Timed Pipeline, Queue Buffer, Load Fluctuation Tolerance

目次

目次

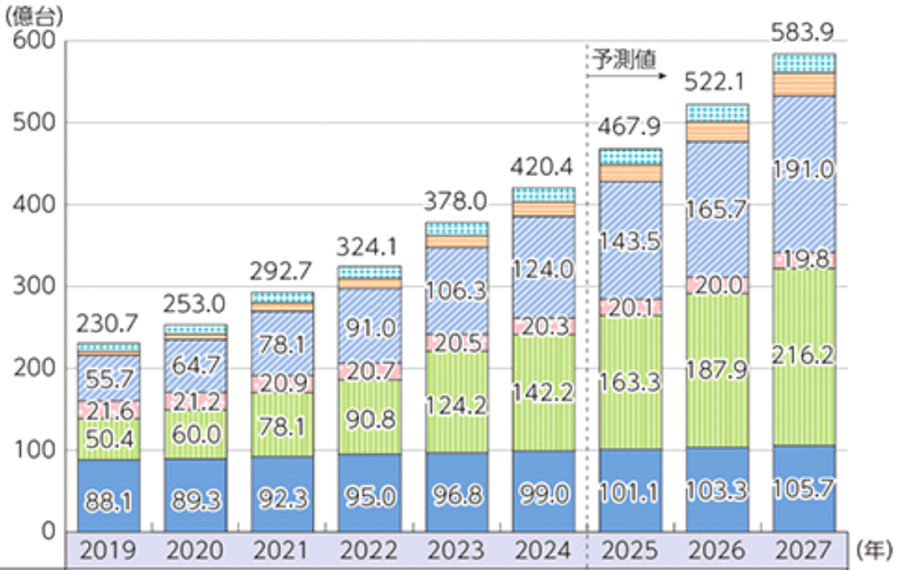
図目次

表目次

第 1 章

序論

近年，IoT(Internet of Things) 技術の普及から，IoT デバイス数は増加傾向にあり，総務省のデータによると，2027 年における世界の IoT デバイス数は，580 億台以上にまで達すると予測されている [1]．中でも工場の自動化などの産業用途や，スマート家電などのコンシューマ用途での需要増加が目覚ましい．



	2019	2020	2021	2022	2023	2024	2025	2026	2027
合計	230.7	253.0	292.7	324.1	378.0	420.4	467.9	522.1	583.9
自動車・宇宙航空	9.8	11.3	12.7	14.2	15.7	17.3	19.1	20.9	22.7
医療	5.1	6.5	10.7	12.5	14.6	17.5	20.7	24.3	28.6
産業用途	55.7	64.7	78.1	91.0	106.3	124.0	143.5	165.7	191.0
コンピュータ	21.6	21.2	20.9	20.7	20.5	20.3	20.1	20.0	19.8
コンシューマ	50.4	60.0	78.1	90.8	124.2	142.2	163.3	187.9	216.2
通信	88.1	89.3	92.3	95.0	96.8	99.0	101.1	103.3	105.7

図 1.1　世界の IoT デバイス数の推移及び予測 (文献 [1] より引用)

これらの用途における IoT システムは，温度や画像など様々なセンサデータを大量に扱う場合が多いため，エッジコンピューティングによる実現が期待されている．従来のクラウドコンピューティングでは，IoT デバイスにより得たデータをすべてクラウドサーバに集約して処理を行うのに対し，エッジコンピューティングは，クラウドサーバにはデータをなるべく送信せず，IoT デバイス上やその近辺に設置されたエッジサーバで処理を行うため，通信遅延を低減することが可能となり，リアルタイム性に優れている [2]．しかし，エッジコンピューティングでは，IoT デバイス上で AI や画像処理など高負荷な処理を行うことが求められ，さらに大量の IoT デバイスが長期間動作することを想定すると，総消費電力量が爆発的に増加してしまうことが問題視されている．以上のことから，IoT デバイスに適したプロセッサの要件として，高性能かつ省電力であることが求められている．

この条件を満たす技術として，データ駆動型プロセッサ (DDP:Data-Driven Processor) が注目されている [3]．DDP はデータ駆動型処理方式に従って処理を行うため，データ依存のない処理を多重並列に実行することが容易であることから，高度な処理性能を有している．さらに，セルフタイム型パイプライン (STP:Self-Timed Pipeline) 回路により構成することで，データの送受が発生した回路のみが駆動し，電力が供給されることから省電力な動作が期待される．

一方，処理性能の面では，周回パイプライン内を巡回するデータパケットの流れを円滑にしないとパイプライン処理性能を充分に発揮できず，最悪の場合，パケット転送が停止する可能性があるといった問題を内在している．この問題に対して，先行研究では，巡回パケット数の変動を吸収可能なキューバッファ機構 QB(Queue Buffer) を DDP パイプライン内に導入する方式が提案されている [4][5]．しかし，これら研究では，どの程度の容量の QB を導入すれば，どの程度の負荷変動耐性を備えられるのかは，十分には明らかにされていない．また，十分な QB 容量が設定されていたとしても，DDP 自体の処理性能を大幅に越える間隔でプログラムの実行が繰り返されると，QB に格納されるパケット数は際限なく増え続け，最終的に負荷変動を吸収しきれなくなる恐れがある．特に，DDP の実用が想定される IoT 用途においては，センサデータのフィルタリング処理のような一定間隔で入力される

データの処理を行うストリーム実行が多いことから，次世代の入力までに処理を完了させるプログラム処理性能を有していることを保証する必要がある．

そこで，本研究では，応用プログラムの実行時の負荷変動状況に基づいて QB の適切な容量およびプログラム実行性能を見積ることによって，負荷変動耐性を備えたデータ駆動型プロセッサを構成する方法を提案した．結果，プログラムの特徴をランク毎に捉えることによって，各 QB 容量およびプログラム実行性能について，誤差は生じるものの，上界を特定できる見通しが得られた．

本論文では, 第 2 章でデータ駆動型プロセッサ (DDP) が採用するデータ駆動型処理方式と，動作を実現するための技術であるセルフタイム型パイプライン (STP) 回路および DDP のステージ構成について述べる．また，負荷変動によって生じる処理性能低下の原因とその対策として設計された QB の回路構成および DDP への導入について述べる．第 3 章では, 負荷変動耐性を担保するための条件となる QB の必要十分容量およびプログラム実行性能の定式化について述べる．第 4 章では，提案した負荷変動耐性条件の見積りモデルについて，回路シミュレーション上で実行した結果との比較評価を行った結果と考察を述べる．第 5 章では，本研究のまとめと今後の課題について述べる．

第 2 章

QB を搭載した DDP 構成

2.1 緒言

本研究では，セルフタイム型パイプライン (STP) 回路により構成された非同期型回路であるデータ駆動型プロセッサ (DDP) について扱う．そこで本章では，DDP の処理方式であるデータ駆動型処理方式および駆動原理である STP について説明し，DDP のステージ構成の詳細を述べる．さらに，DDP における課題として，巡回パケット数の変動による処理性能の低下について述べ，その解決策となる DDP への QB 回路の導入について述べる．

2.2 データ駆動型処理方式

DDP は，データ駆動型処理方式に従って演算を行うプロセッサである [3][6]．データ駆動型処理は図 2.1 に示すように，演算をノード，データ依存関係を有向辺 (アーク) で表したデータフロー図により表される．演算対象となるデータは，アークを通ってノードに入力され，各ノードはその演算に必要なデータが揃った場合に発火し，演算結果を出力する．具体的なデータ駆動型処理の例として入力 A,B,C,D に対して $(A+B) \times (C-D)$ を計算するプログラムを表したデータフロー図を図 2.2 に示す．ここで，node0, node1 間にデータ依存関係が無いため，それぞれ並列に処理をおこなうことが可能である．一方，node2 は node0，node1 から出力されたデータを基に演算を行うため，データ依存関係があり，それらの演算結果が出力された後に処理が行われる．このように，データ依存関係のない演算を自然に並列実行することが可能であるため，一般的なプロセッサで採用されている逐次実行

処理方式よりも多重並列性に優れている．

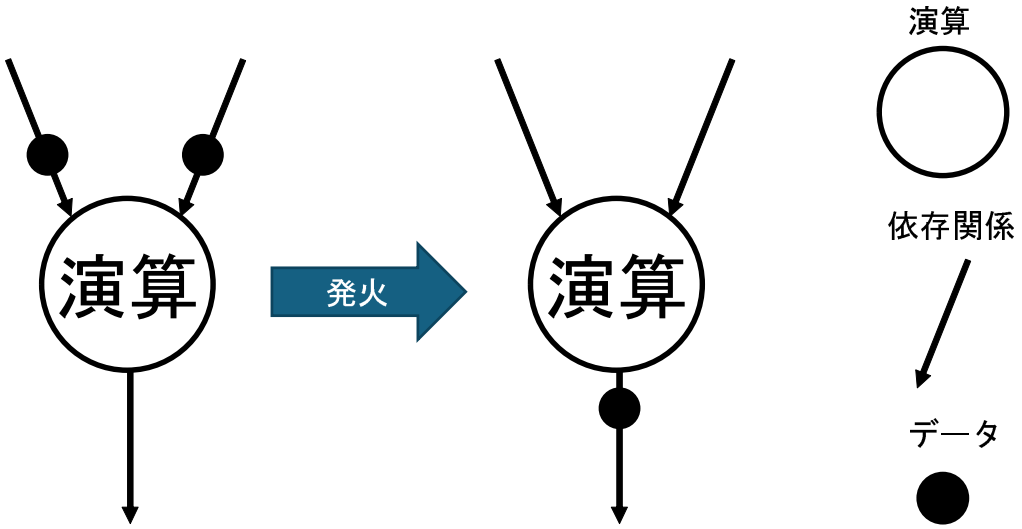


図 2.1　データ駆動型処理方式の動作原理

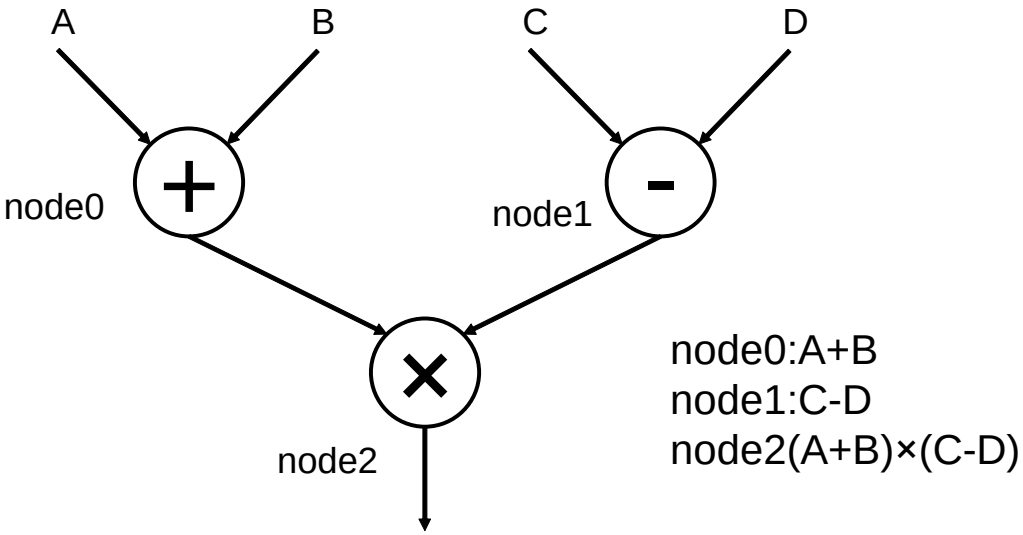


図 2.2　データ駆動型プログラムの実行例

2.3　セルフタイム型パイプライン STP

本研究で扱う DDP は，セルフタイム型パイプライン (STP) 回路により構成された非同期型回路である．STP 回路は，図 2.3 に示すように，前段ステージから転送されたデータを

保持するデータラッチ (DL)，データ転送の制御を行う C 素子 (Coincidence flip-flop)，そして各ステージに応じた処理を行うデータ処理回路 (Logic) を合わせて 1 ステージとし，複数段にわたってステージが接続されることで構成されている．本研究で扱う C 素子は 4-phase ハンドシェイクによる転送制御を採用しており，各ステージの C 素子は，前段ステージの C 素子からのデータ転送要求信号 (Send) を受けると，前段ステージの C 素子に対してデータ転送許可信号 (Ack) を返し，前段ステージの C 素子は Ack を受けると，再度 Send を送るため，2 度目の Send を受け取った C 素子は DL 更新信号 (CP) を発信する．CP の立ち上がりによって，DL は前段ステージの Logic で処理されたデータを取り込み，同様の手順で，後段ステージにデータを転送する．後段ステージへの転送が完了すると，前段ステージの C 素子に対して 2 度目の Ack を送り，前段ステージとの転送を完了する．このように隣接するステージの C 素子間のハンドシェイクによりデータ転送を制御することで，データの処理が必要な箇所の回路のみが駆動することになるため，処理を行わない回路の待機時電力を削減することができ，省電力な動作を実現することが可能となる．

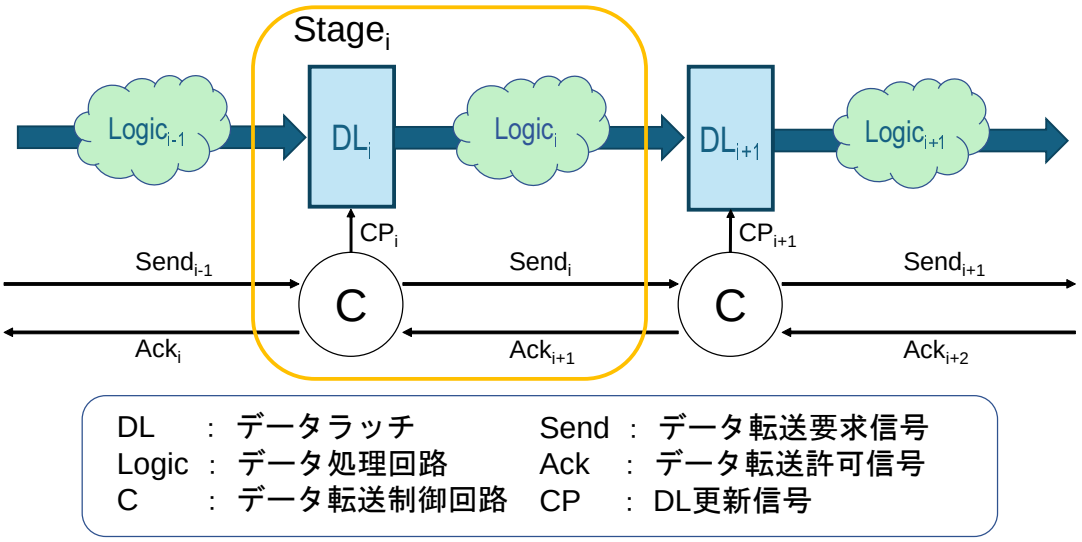


図 2.3　STP の動作原理

2.4　DDP のパイプラインの基本構成

本研究で扱う DDP は STP により構成され，図 2.4 で示すように，基本的構成要素 8 ステージを環状に接続し，各ステージが異なる処理を行うことで図 2.1 で示したデータ駆動型処理を実現する [6]．DDP では，図 2.5 に示すように，入力されるデータとして，各ステージにおける処理で必要な情報であるヘッダと処理を行う対象であるデータをまとめたパケットという形で与えられる．各ステージにおける処理で必要になる情報は異なってくることから，パケットが有する情報も異なる．ここでは例として，外部から入力されるパケットが有している情報を表 2.1 に示す．各ステージでは，前段ステージからの入力パケットが有する情報を基に処理を行い，パケットの情報を更新して後段ステージへと出力する．以下では，各ステージの詳細について説明する．

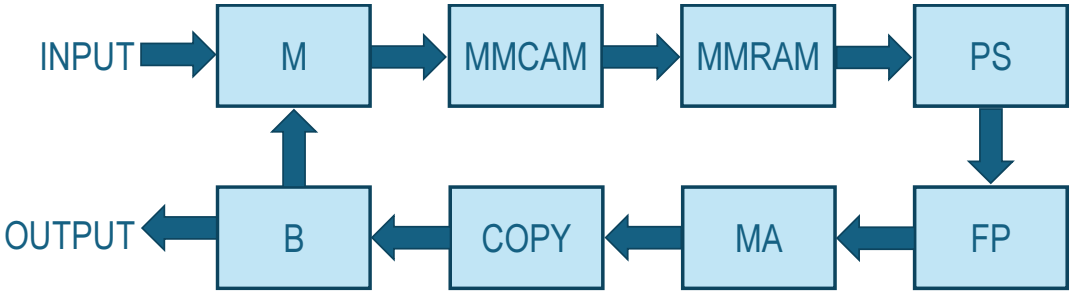


図 2.4　DDP のステージ構成

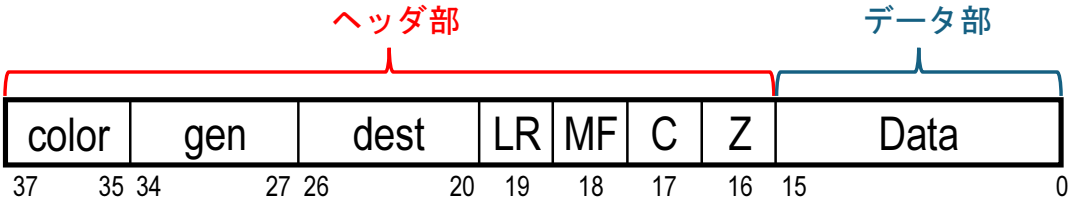


図 2.5　入力パケットの例

表 2.1　入力パケット情報の例

フィールド名	bit 数	情報
color	3bit	インスタンス識別子
gen	8bit	パケットの世代
dest	7bit	パケットの宛先
LR	1bit	入力方向
MF	1bit	マッチングフラグ
C	1bit	キャリーフラグ
Z	1bit	ゼロフラグ
Data	16bit	データ

2.4.1　合流調停 (M ステージ)

M ステージでは，外部から入力されたパケットと DDP パイプライン上を巡回してきたパケットの合流を調停する．制御信号として外部からの Send と B ステージからの Send が入力され，片方の転送が完了するまでもう一方のパケット転送を停止することによりパケットの転送を調停する．また，本研究で扱う DDP では，外部からの入力を優先して調停を行う．

2.4.2　パケット待ち合わせ (MMCAM ステージ)

MMCAM ステージでは，転送されてきたパケットが二項演算である場合にマッチングを行う．入力されたパケットが二項演算であり，対となるパケットが到着していない場合には，到着するまで待機しておく必要があるため，マッチングに必要なパケットのヘッダ情報を一時的に連想アドレッシングメモリ CAM(Content Addressable Memory) に格納し，MMRAM ステージにパケットおよび転送停止信号を出力する．入力されたパケットが定数演算, または二項演算で対となるパケットが到着していた場合には発火し，転送を行う．

2.4.3 データ統合 (MMRAM ステージ)

MMRAM ステージでは，マッチング中のパケットのデータや定数データが格納され，マッチングの発火時および定数演算時にデータを統合し，PS ステージにパケットを転送する．

2.4.4 命令読出 (PS ステージ)

PS ステージでは，パケットが持つヘッダ情報を基に，実行する命令の演算識別子 (OPC) を全プログラムが保存されたプログラムストレージ (PS) から dest をアドレスとして読み出してパケットに付加し，FP ステージに転送する．

2.4.5 演算実行 (FP ステージ)

FP ステージでは，パケットが保持している OPC を基に，対応する演算を行う．算術演算や論理演算の場合にはパケットが保持している二つのデータに対して行った演算結果を付加し，ロード命令やストア命令の場合にはデータメモリにアクセスするためのアドレスを算出したデータを付加して，MA ステージに転送する．

2.4.6 データアクセス (MA ステージ)

MA ステージでは，ロード命令やストア命令である場合にデータメモリ (DMEM) にアクセスし，データの書き込みおよび読み出しを行う．ロード命令の場合には，DMEM から読み出したデータをパケットに付加して COPY ステージに転送し，ストア命令の場合には，入力されたパケットに付加されたデータを DMEM に書き込み, それと同じデータをパケットに付加して COPY ステージに転送する．また，それ以外の命令はデータアクセスが不要であるため，何もデータを付加せずにパケットを COPY ステージに転送する．

2.4.7　パケット複製 (COPY ステージ)

COPY ステージでは，パケットに応じて，パケット複製が必要である場合にはパケットを複製し，B ステージに転送する．また，本研究で扱う DDP では，複製されたパケットは複製元のパケットの次の行先ノードである dest に対して+1 されたものを付加する．

2.4.8　分岐 (B ステージ)

B ステージでは，パケットに応じて，外部に転送する，あるいは再度巡回し，M ステージに出力するかを判別し分岐制御を行う．

2.5　DDP における負荷変動による処理性能低下とその対策

プロセッサの処理性能として，単位時間あたりに実行可能な演算数である OPS(Operations Per Second) が用いられている．一般的な同期回路の処理性能は大域的クロックごとに演算が実行されるため，クロック周波数から求めることが可能である．それに対して，非同期型回路である DDP は，隣接するステージの C 素子間のハンドシェイクにより発信される CP の立ち上がりで処理が実行される．CP の立ち上がりの周期は，図 2.6 に示すように，ステージ i において CP_i が立ち上がってから後段ステージの CP_{i+1} が立ち上げることができる時間として転送時間 T_{f_i}，後段ステージの CP_{i+1} が立ち上がってから，再度 CP_i が立ち上がることができる時間を転送許可時間 T_{r_i} として定義され，それらを合わせた転送制御遅延時間 $(T_{f_i} + T_{r_i})$ を要する．ただし，転送制御遅延時間は各ステージの処理内容によって異なり一律ではない．さらに DDP におけるパケット転送は一定間隔で発生するわけではないため，同期回路と同様の手法で処理性能を求めることが困難である．そのためここでは，1 パケットがステージを通るごとに処理が施され，1 周することで 1 ノード分の演算が実行されることから，1 秒間あたりにパイプラインを周回するパケット数を OPS と定義して議論を進める．

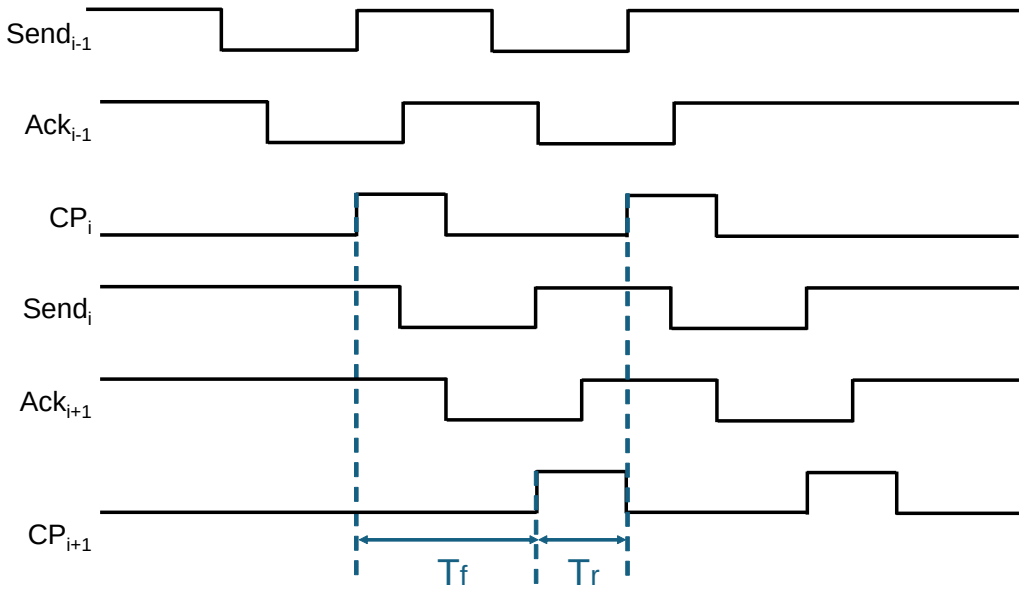


図 2.6　C 素子間でのハンドシェイクのタイミングチャート

パイプラインステージ数を pl，同時に巡回するパケット数を N，パケットが 1 周するのに要する時間を T_l, 最も転送時間が長いステージの転送制御遅延時間を最大転送制御遅延時間 $T_{max} = \max(T_{f_i} + T_{r_i})$ とおく．この時，1 秒間あたりにパイプライン上を周回するパケット数は $\frac{N}{T_l}$ となり，$OPS = \frac{N}{T_l}$ を得る．そのため，T_l が一定であると仮定すると，OPS は巡回パケット数に比例して向上するため，処理性能の面では，可能な限り多数のパケットを同時に巡回させることが望ましいといえる．ただし，T_l は，巡回パケット数 N により変動する [7]．DDP では，パケットが連続して転送される場合，後段ステージのパケットが転送されるまで次のパケットを転送することができないため，T_{max} により律速される．このようなパケット転送の輻輳により生じた渋滞の伝搬は，後続パケットの転送間隔の余裕により緩衝することが可能である．しかし，巡回パケット数が多く，パケットの転送間隔に余裕がない場合には，渋滞の伝搬を緩衝しきれず，次周のパケット転送にまで影響が及ぶため，T_l が増加し，処理性能の低下につながる．さらに，$N = pl$ かつ，B ステージのパケットが再度周回する場合には，すべてのステージにパケットが保持され，すべてのステージが転送できなくなることによるデッドロックが発生する．

以上のように，DDP の処理性能を向上させるためには，単純に巡回パケット数を増やせば良いわけではなく，巡回パケット数を一定に保つことで円滑なパケットの流れを維持し，これらの問題が生じないような工夫が必要となる．現状，この問題に対する解決策として，ソフトウェア的手法とハードウェア的手法が提案されている．

2.5.1　ソフトウェア的手法

ソフトウェア的手法では，プログラム構造を変形することにより，パケットの入力やパケットの複製などの巡回パケットの増加要因と，パケットの出力やマッチングなどの減少要因の発生を調整することにより，巡回パケットを一定に保つ．しかし，この手法では，プログラムが複雑化してしまうことや，最適な巡回パケット数を維持し続けることが困難であるといった問題がある．また，DDP の処理では，同時に複数のプログラムを実行することも想定されるため，その場合，巡回パケット数はそれぞれのプログラムを重ね合わせたものとなることから，プログラムの工夫のみで解決することは難しい．

2.5.2　ハードウェア的手法 (QB 機構)

ハードウェア的手法では，後段ステージが詰まっている場合に，一時的にパケットを退避させることで輻輳を解消するキューバッファ(QB) 機構の導入が提案されている [4][5]．従来の DDP における各ステージでは，パケットが一つ転送されると，そのパケットが後段ステージに転送し終わるまで次のパケットを受け付けない．それに対して，QB では後段ステージにパケットを転送できない状態であっても，キューに空きがある限りはパケットを受け付け，転送されてきた順に転送を行っていく．そのため，QB におけるパケット転送の制御は後段とのハンドシェイクに依存せず，前段とのハンドシェイクを行うことが可能となり，前段へのパケット転送の渋滞を解消できる．

この手法について，先行研究 [4] では，パケット転送の輻輳が発生する箇所に対して，QB 機構の適用が検討されており，ソフトウェア上での離散事象シミュレーションによる評価の

結果，輻輳が解消されたことにより処理性能が向上することが確認された．またこの研究では，パイプライン段数に依存しない QB 容量の十分量の推定が行われている．しかし，この手法ではプログラム構造や DDP アーキテクチャ特性に基づく動的な負荷変動を考慮できていないため，適応する応用システムによっては非常に冗長に見積ってしまう恐れがある．また，先行研究 [5] では，非同期 FIFO を利用した QB 回路が設計され，実際に QB 機構を導入した DDP の配置配線後の実遅延回路シミュレーションでの実証により，DDP における QB 機構の有用性が確認された．しかしこの研究では，どの程度の容量の QB を導入すれば，どの程度の負荷変動耐性を備えられるかは，明らかにされておらず，導入する QB のバッファ容量が多すぎる場合には，冗長な回路規模の増大につながり，逆に少なすぎる場合には負荷変動を吸収しきれずに処理性能の低下が生じてしまうといった問題が残されている．また，十分な QB 容量が設定されていたとしても，DDP 自体の処理性能を大幅に越える間隔でプログラムの実行が繰り返されると，QB に格納されるパケット数は際限なく増え続け，最終的に負荷変動を吸収しきれなくなる恐れがある．特に，DDP の実用が想定される IoT 用途においては，センサデータのフィルタリング処理のような一定間隔で入力されるデータの処理を行うストリーム実行が多いことから，次世代の入力までに処理を完了させるプログラム処理性能を有していることを保証する必要がある．

2.6　QB 回路の構成と DDP への導入

DDP に対して QB 機構を導入したパイプライン構成を図 2.7 に示す．本研究では，パケット転送の輻輳が生じる箇所として入力パケットと巡回パケットの衝突が発生する箇所となる M ステージの前段に QB1，パケットの複製が発生する箇所となる COPY ステージの前段に QB2 として新たなパイプラインステージを追加した．また，先行研究 [5] にて導入された QB 回路のモジュール構成を図 2.8 に示す．

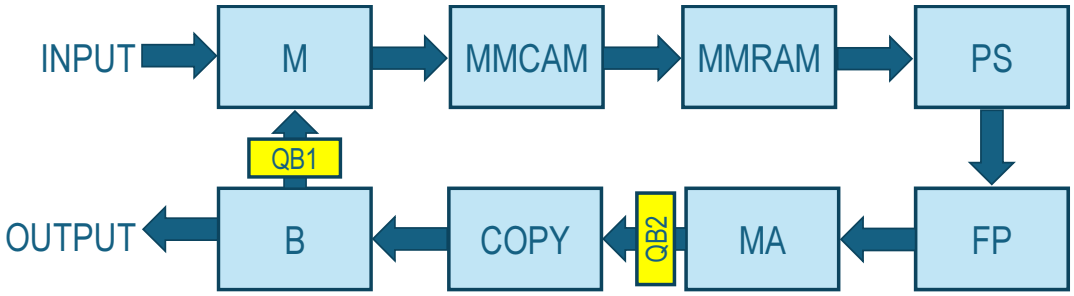


図 2.7　QB を搭載した DDP パイプライン構成

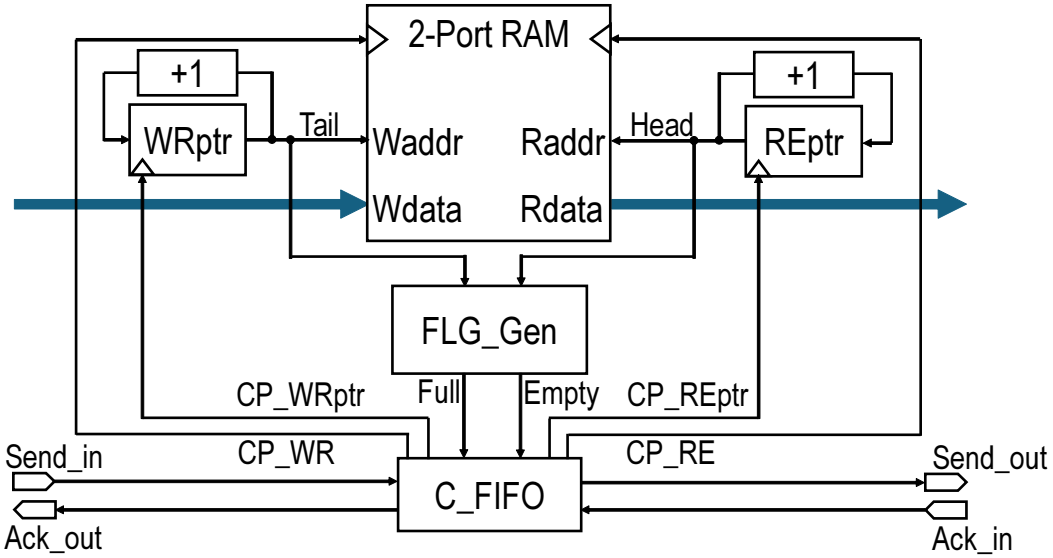


図 2.8　QB の構成

2.6.1　2-Port RAM

2-Port RAM は，キューにおけるデータを格納するメモリとしての役割を持つ．本研究では，AMD 社製 FPGA 用 EDA ツール Vivado で用意された IP(Intellectual Property) である BlockRAM を使用する．QB は書き込み，読み出しのタイミングが異なることから，それぞれ別々のクロックを用いることができるシンプルデュアルポート RAM を用いる．書き込み，読み出しのためのクロック信号はそれぞれ，C_FIFO から送られてくる信号である CP_WR，CP_RE を用いる．RAM のメモリサイズは QB 容量 (保持可能なパケット数) を

M，保持するパケットの長さを S とすると $M * S$bit 分の容量が必要になる．ここで，DDP のパケットサイズはステージごとに異なってくるため，QB を適用する箇所を通るパケットのサイズに合わせる必要がある．

2.6.2　WRptr, REptr

WRptr，REptr は，それぞれ Tail(次に書き込みを行うアドレス) と Head(次に読み出しを行うアドレス) を保持するためのポインタである．ポインタの容量は，メモリアクセスを行う際のアドレスとして利用されることから，最低でも QB 容量分のパケットを識別可能な bit 数が必要であり，かつ後述の FLG_Gen においてアドレスに加えて余分に 1bit 必要になることから，$\lceil \log_2 M \rceil + 1$ bit 必要になる．それぞれのポインタの更新は，C_FIFO から送られてくる信号である CP_WRptr，CP_REptr の立ち上がりで値の更新を行う．

2.6.3　FLG_Gen

FLG_Gen では，Head，Tail の値を基に，キューの状態を表すフラグである Empty(キューが空の状態)，Full(キューが満タンの状態) の生成を行う．ここで，ポインタのサイズをアドレス (Head，Tail) の長さとした場合の Empty，Full フラグの判定条件は，

$$Empty = (WRptr == REptr)$$

$$Full = (WRptr == REptr)$$

のように Empty と Full の条件が同じになり，判別することが出来ない．そのため，それぞれのポインタにアドレスの長さに加えて余分に 1bit 付与することにより，

$$Empty = (WRptr == REptr)$$

$$Full = (WRptr(Tail) == REptr(Head))\&(WRptr(MSB) != REptr(MSB))$$

のように，Full の場合には一周多くまわり，MSB が反転するため，Empty と Full を判別することが可能となる [8].

2.6.4 C_FIFO

C_FIFO は，QB と前後のステージ間の転送制御を行う C 素子である．通常の C 素子と異なり，前段のハンドシェイクと後段のハンドシェイクは独立して行われ，キューが Full でなければ，後段の転送状況によらず前段とのハンドシェイクを行い，キューへの書き込み信号 CP_WR とポインタの更新信号 CP_WRptr を生成する．また，キューが Empty でなければ，前段との転送状況によらず後段とのハンドシェイクを行い，キューからの読み出し信号 CP_RE とポインタの更新信号 CP_REptr の生成を行う．

2.7 結言

本章では，DDP における問題を提起するにあたり，DDP の処理方式であるデータ駆動型処理方式，駆動原理である STP，そして DDP のパイプライン構成について述べた．そして，DDP における問題として，巡回パケット数の変動による処理性能の低下を挙げ，その解決策として提案された QB 機構について述べた．

第 3 章

負荷変動耐性条件の見積りモデル

3.1 緒言

本章では，初めに負荷変動耐性を担保する条件についての定式化を行うにあたっての方針を述べる．続いて，プログラム実行時の負荷変動状況について増加減少の要因を整理し，プログラム構造および DDP アーキテクチャ構成から簡易的に定量化を行う方法について述べる．そして，その結果に基づいて負荷変動耐性を担保するための条件となる必要十分な QB 容量とプログラム実行性能の見積りモデルを提案する．

3.2 定式化の方針

データ駆動型プログラム実行時における負荷変動の状況は，実行される応用プログラムのプログラム構造やノード間の依存関係，プログラムの実行レート，さらには実行基盤となる DDP のアーキテクチャ構成など，複数の要因によって大きく異なる．そのため，システム設計者には，これらの要因を総合的に考慮したうえで，負荷変動に対して十分な耐性を有するシステム設計を行うことが求められる．ここで，従来の設計手法においては，回路シミュレーションや機能レベルシミュレーションによる動作検証が一般的に用いられてきた．これらの手法では，すべてのステージ間におけるデータ転送や制御の挙動を逐一，詳細に模擬する必要があるため，大規模なプログラムや複雑なアーキテクチャを対象とした場合には，シミュレーションの実行に多大な時間を要するという課題がある．さらに，回路シミュレーションは回路実装後にしか実行できないため，設計初期段階での評価が困難となり, システ

ムが所望の性能や負荷変動耐性といった要件を満たしていないことが判明した場合には，回路設計のみならず，場合によってはシステム全体の要件定義にまで立ち戻る必要が生じる．このような手戻りは設計期間の長期化を招き，時間的および人的コストの観点からも極めて大きな負担となる．以上のことから，DDP 実行時における負荷変動状況をより簡易的かつ抽象的に定量化し，設計の早期段階において負荷変動耐性を担保する条件を見積る手法が強く望まれている．そこで本研究では，図 3.1 のように，入力から同じ遷移回数で到達可能なノード集合を同一周において並列に実行されると仮定し，プログラム中におけるランク集合として定義することにより，このランク集合の遷移に基づいてプログラムの負荷変動状況を簡易的に定量化する．さらにその結果に基づいて，負荷変動耐性を担保するための条件となる必要十分な QB 容量とプログラム実行性能の見積りモデルを提案する．

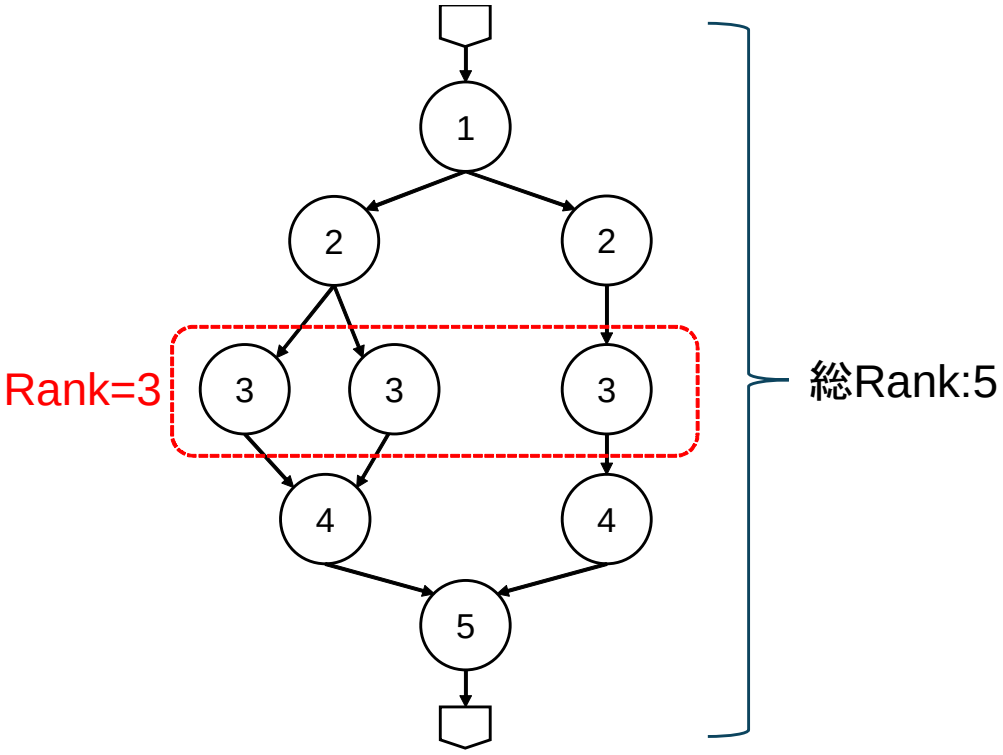


図 3.1　データ駆動型プログラムのランク集合

3.3　負荷変動状況の定量化

データ駆動型処理に基づく DDP でのプログラム実行では，プログラムの実行状況や巡回するパケット数などが時間的に変化するため，DDP 内部の負荷も動的に変動する．この負荷変動は，処理性能の低下やデッドロックを引き起こす要因となることから，安定したシステム動作を実現するためには，その発生要因を定量的に把握することが重要である．しかし，負荷変動はプログラム構造やアーキテクチャ特性など複数の要因が複雑に関係して生じるため，直感的な議論のみで評価することは困難である．

そこで本節では，DDP における負荷変動状況を解析的に扱うことを目的として，まずプログラム構造に起因する負荷変動要因を抽出し，それらを定量化する．さらに，DDP アーキテクチャの特徴についても定量的に考慮することで，以降の節で負荷変動耐性条件を定式化するための基盤を構築する．

3.3.1　プログラム構造の定量化

DDP で実行されるデータ駆動型プログラムは，図 3.2 に示すように，複数のパターンによってノード間が接続されることで構成されており，これらの組み合わせに応じて負荷変動の状況が変化する．そのため，本研究では，これらのパターンのうち負荷変動の増加要因および減少要因を整理し，それぞれについてランク k における要素数を列挙することにより，プログラム構造の定量化を行う．

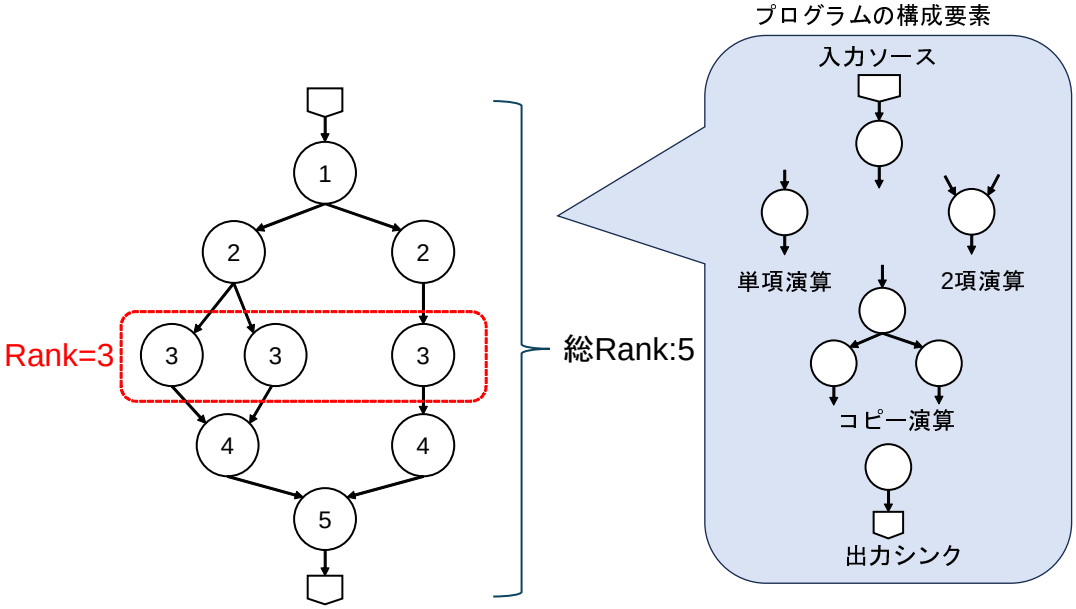


図 3.2　データ駆動型プログラムの構成要素

まず，すべてのデータ駆動型プログラムは，外部からパケットが入力されることを契機として実行が開始されるため，一つ以上の入力ソースを有する．入力ソースでは外部からパケットが入力されることにより巡回パケット数が増加するため，負荷変動の増加要因となる．そこで，ランク k における入力ソース数を N_{in_k} と定義する．

次に，DDP において処理された結果は外部へ出力されるため，プログラムは一つ以上の出力シンクを有する．出力シンクは外部へパケットを出力する役割を担うことから，巡回パケット数を減少させる要因となる．したがって，ランク k における出力シンク数を N_{out_k} と定義する．

さらに，実行されるプログラムの内容によっては，同一のオペランドを複数回使用する場合があり，その際にはパケットの複製が行われる．パケットの複製が行われると，一つのパケットを 2 回転送するため，巡回パケット数が増加し，負荷変動の増加要因となる．そこで，ランク k におけるコピー演算数を N_{copy_k} と定義する．

また，二つのデータを入力として処理を行う二項演算では，二つのパケットを統合して演算を行い，その結果として一つのパケットを出力する．この処理は巡回パケット数を減少さ

せる要因となるため，ランク k における二項演算数を N_{match_k} と定義する．

最後に，パケットはノード間を接続するアークをたどって遷移することから，巡回パケット数は当該ランクにおける出力アーク数と等価である．よって，ランク k における出力アーク数を，ランク k における巡回パケット数 N_k として定義する．

次節では，これら 5 つのプログラム構造のモデルを基に負荷変動耐性条件の定式化を行う．ただし，現実的なデータ駆動型プログラムでは，条件によって行先ノードが変わる分岐構造や複数回同じノードをたどるループ構造を持つ．こうした構造は，プログラム構造だけでなく，入力データによっても実行状況が変化するため，一般化することが困難である．そのため，本研究では，そうした構造について最も負荷が大きくなるような状況を想定した定量化を行う．具体的には，実行プログラムが分岐構造を持つ際には図 3.3 のように，分岐以降の部分グラフを比較し，巡回パケット数 N_k が最大のものを有する部分グラフを採用する．また，実行プログラムがループ構造を持つ場合には図 3.4 のように，システム要件で設定された最大のループ回数で展開する．

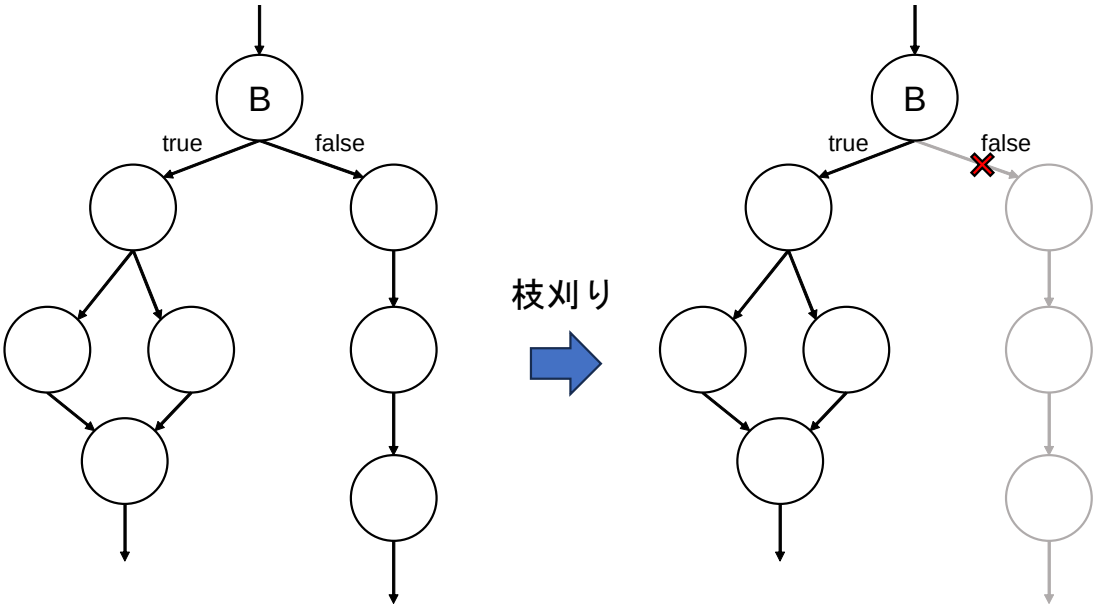


図 3.3　分岐構造への対応 (枝刈り)

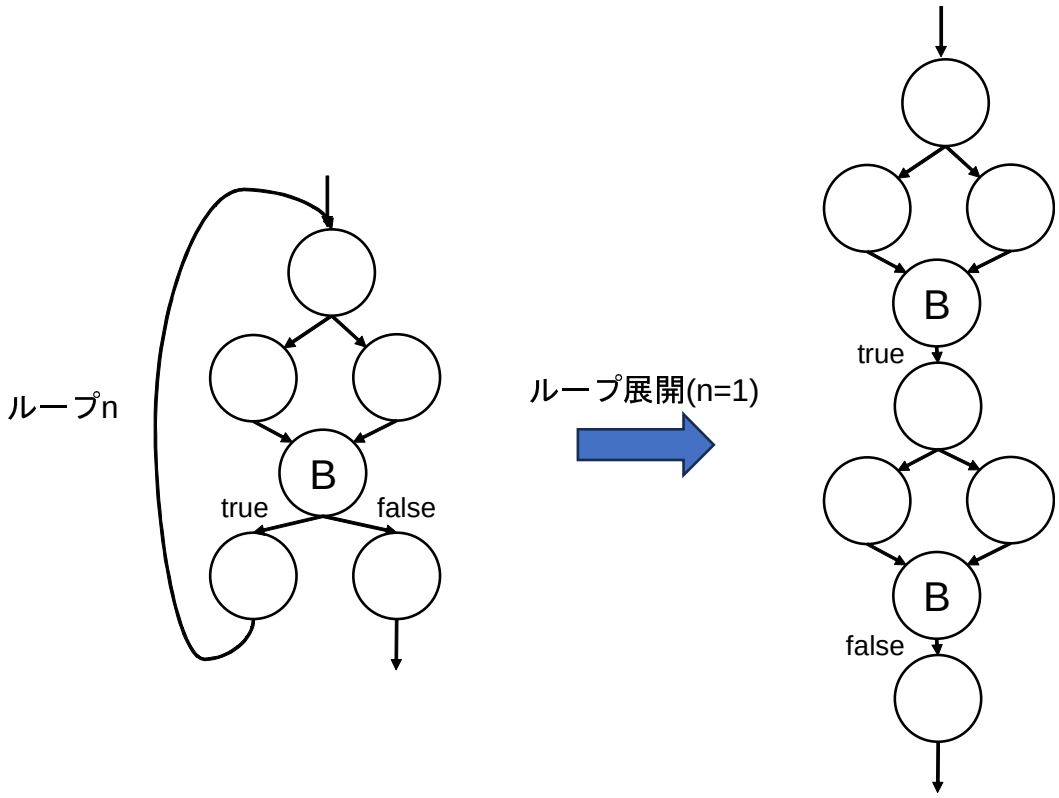


図 3.4　ループ構造への対応 (ループ展開)

3.3.2　DDP アーキテクチャの定量化

DDP では，各ステージの処理内容によってそのステージが転送に要する時間である転送時間 T_f, 転送許可時間 T_r が異なる．ここで，ステージ i における転送時間を T_{f_i}, ステージ i における転送許可時間を T_{r_i} と定義する．そのため，ステージ i では，それらを足し合わせた転送制御遅延時間 $(T_{f_i} + T_{r_i})$ の間隔で転送が行われる．ただし，連続してパケットが転送される場合，後段の転送が完了するまで転送することができないことから，最大転送制御遅延時間 $T_{max} = \max(T_{f_i} + T_{r_i})$ により律速を受けるため，連続するパケットの転送間隔は T_{max} 以上で保たれる．ただし，QB1 から QB2 への転送経路と QB2 から QB1 への転送経路は，QB を介して分断されており，それぞれの転送は QB により緩衝されることから相互に影響を及ぼさない．そのため，それぞれの転送経路に対して個別に律速条件を考える必要がある．そこで，QB1 から QB2 への転送経路における最大転送制御遅延時間を T_{max1},

QB2 から QB1 への転送経路における最大転送制御遅延時間を T_{max2} と定義する．ここで，本研究で対象とする DDP パイプラインでは，QB1 から QB2 への転送経路内に MMCAM ステージなど処理時間の大きいステージが含まれる一方で，QB2 から QB1 への転送経路内のステージは比較的処理時間が小さい．そのため，$T_{max} = \max(T_{f_{mmcam}} + T_{r_{mmcam}})$，$T_{max1} > T_{max2}$ が成り立つ．次節ではこれらの条件のもと議論を進める．

3.4　必要十分な QB 容量の見積りモデル

必要十分な QB 容量の見積りモデルを構築するにあたり，本研究では二つのアプローチを採用する．一つ目のアプローチでは，プログラムの最大巡回パケット数のみに着目し，DDP 内部の挙動を詳細には考慮せず，マクロな観点からモデル化を行う．この方法では，QB1 および QB2 それぞれの増減要因を区別しないため，両者を合算した総 QB 容量に対する必要条件を導出することができる．

しかしながら，当該必要条件を満たしている場合であっても，各 QB へのパケット蓄積に偏りが生じた場合には，DDP 全体の処理性能が低下する可能性がある．そこで二つ目のアプローチとして，各 QB の増加・減少要因に着目し，ランクごとにプログラムの実行状況を簡易的に模擬することで，各 QB について十分な容量を個別に見積るモデルを構築する．

3.4.1　必要条件

巡回パケット数に応じたパイプライン周回時間 T_l を考える [7]．パイプライン上を巡回するパケットは，転送時間が最大であるステージ（以下，T_{max} を要するステージ）によって律速される．このため，T_{max} 以降のステージにおける連続するパケットの転送間隔は T_{max} 以上に保たれる．

また，パケットの転送間隔が T_{max} 以上確保されている場合には，いずれのステージにおいても Send の到着前に Ack が到達している．その結果，Ack 待ちによる律速は発生せず，T_r の影響を無視することができる．このとき，すべてのステージ i における転送は純粋な

転送時間 T_{f_i} のみで完了する．したがって，DDP パイプラインを 1 周するのに要する時間は，各ステージの T_{f_i} を積算した

$$T_l = \sum T_{f_i} \tag{3.1}$$

となる．この状態は，すべてのパケットの転送間隔が T_{max} 以上である場合に成立する．したがって，巡回パケット数が $\frac{\sum T_{f_i}}{T_{max}}$ 以下であれば，全パケットの転送間隔が T_{max} 以上に保たれるため，

$$0 \leq N_k \leq \frac{\sum T_{f_i}}{T_{max}} \tag{3.2}$$

を満たす範囲で成立する．

次に，巡回パケット数が増加し，$\frac{\sum T_{f_i}}{T_{max}}$ を超えた場合について考える．この場合，いずれかのパケット間の転送間隔が T_{max} 未満となり，T_{max} を要するステージによって律速が生じる．その結果，当該ステージの直前でパケットの渋滞が発生し，待ち時間が生じる．

この待ち時間は，$\frac{\sum T_{f_i}}{T_{max}}$ を超過したパケット 1 個あたり T_{max} ずつ増加するため，パイプライン周回時間は

$$T_l = \sum T_{f_i} + T_{max} * (N_k - \frac{\sum T_{f_i}}{T_{max}}) \tag{3.3}$$

となる．また，この渋滞は巡回パケット数が減少しない限り持続する．

ただし，T_{max} を要するステージの前段ステージ i における本来の転送時間は $(T_{f_i} + T_{r_i}) \leq T_{max}$ を満たすため，$(T_{max} - (T_{f_i} + T_{r_i}))$ の余裕時間を有している．この余裕により，渋滞の伝搬は緩衝され，スループット低下には至らない．

さらに，本研究では QB を導入したことにより，キューに空きがある限り渋滞の伝搬が緩衝される．QB が有する緩衝能力は $T_{max} * (QB$ 容量$)$ である．よって，本研究で扱う DDP は最大 $\sum (T_{max} - (T_{f_i} + T_{r_i})) + T_{max} * (QB1 + QB2)$ の緩衝能力を有している．ただし，$QB1$ は QB1 の容量，$QB2$ は QB2 の容量である．この状態は，

$$T_{max} * (N_k - \frac{\sum T_{f_i}}{T_{max}}) \leq \sum (T_{max} - (T_{f_i} + T_{r_i})) + T_{max} * (QB1 + QB2) \tag{3.4}$$

を満たすとき成立する．

さらに，巡回パケット数が増え，式 (3.4) を満たさなくなった場合を考える．この時，T_{max} を要するステージにより律速を受け，直前のステージで発生していた渋滞はパイプライン上を一周し，T_{max} 自身の転送にまで及ぶこととなり，転送のたびに Ack 信号が伝搬する時間分の遅延が生じ，処理性能が極端に低下する．以上のことから，処理性能が低下しないための QB 容量の必要条件は，式 (3.4) より，

$$N_{max} - pl + \frac{\sum T_{r_i}}{T_{max}} \leq QB1 + QB2 \tag{3.5}$$

となる．ただし，N_{max} は同時に巡回する最大パケット数である．

さらに，巡回パケット数が $N_k = pl + QB1 + QB2$ に達すると，すべてのステージにパケットが保持され，すべてのステージが転送できなくなることによるデッドロックが発生する．よって，デッドロックが発生しないための QB 容量の必要条件は，

$$N_{max} - pl < QB1 + QB2 \tag{3.6}$$

となる．本研究では，式 (3.5) を QB 容量の必要条件として採用する．

一方，実際のプログラム実行においては，巡回パケット数の増加要因として外部入力およびコピー処理が存在する．これらが連続して発生した場合，それぞれ M ステージ前および COPY ステージ前で輻輳が生じ，局所的な渋滞が発生する．特に，コピー処理が連続した場合には，QB1 の状態に依存せず，パケットが QB2 にのみ蓄積される．その結果，式 (3.5) を満たしていたとしても，QB2 が溢れることでスループットが低下する可能性がある．そのため，各 QB ごとの最大パケット蓄積量を個別に見積る必要がある．

3.4.2　十分条件

各 QB ごとの最大パケット蓄積量を個別に見積るにあたり，それぞれの QB に蓄積される待機時間の増加減少の要因について整理し，ランク k での QB1 の累積待機時間 T_{QB1acc_k} および QB2 の累積待機時間 T_{QB2acc_k} に寄与する増加量，減少量の定式化を行う．

まず，QB1 において，あるパケットが待機する時間は，外部からの入力により増加する

待機時間 Inc_{in} と，外部への出力により減少する待機時間 Dec_{out1} の差分が主要因となる．

外部からの入力パケットと，パイプラインを周回してきたパケットが同時に M ステージへ到達した場合，衝突が発生し，周回してきたパケットに待ち時間が生じる．このとき，巡回パケットは外部入力パケットの転送が完了するまで転送できないため，T_{max1} の転送遅延が発生する．よって，外部入力が N_{in_k} 回連続した場合，QB1 における待機時間の増加量は

$$Inc_{in} = T_{max1} * N_{in_k} \tag{3.7}$$

となる．

次に，B ステージにおいて外部への出力が発生した場合を考える．外部出力が 1 回発生すると，T_{max2} 間隔で QB1 に入力されていたパケットが 1 つ分欠落するため，QB1 内のパケットの待機時間は T_{max2} だけ短縮される．ただし，QB1 に蓄積されている待機時間を超えて減少することはできないため，前ランクにおける QB1 の累積待機時間 $T_{QB1acc_{k-1}}$ を上限として，

$$Dec_{out1} = \min(T_{QB1acc_{k-1}}, T_{max2} * N_{out_k}) \tag{3.8}$$

となる．

次に，QB2 内での待機時間は，主にコピーに伴い増加する時間とマッチングに伴い減少する時間の差分が主要因である．COPY ステージにおいてパケットのコピーが発生すると，1 つの入力パケットに対して 1 回分多くの転送処理が必要となるため，T_{max2} の転送制御遅延が発生する．よって，コピーが N_{in_k} 回発生した場合，QB1 における待機時間の増加量は

$$Inc_{copy} = T_{max2} * N_{copy_k} \tag{3.9}$$

となる．

一方，マッチングが発生すると，T_{max1} 間隔で QB2 に入力されていたパケットが 1 つ分欠落することになるため，QB2 内のパケットの待機時間は T_{max1} 短縮される．ただし，QB2 に蓄積された待機時間以上に減少することはできないため，前ランクにおける累積待機時間 $T_{QB2acc_{k-1}}$ を上限として，

$$Dec_{match2} = \min(T_{QB2acc_{k-1}}, T_{max1} * N_{match_k}) \tag{3.10}$$

減少する．

さらに，一方の QB が空である場合，もう一方の QB における待機時間の減少要因として作用するため，QB 間の相互作用を考慮する必要がある．

QB1 については，マッチング処理が発生した際に QB2 が空である場合，T_{max1} 間隔のパケット 1 つ分が欠落することになるため，QB1 の待機時間は

$$Dec_{match1} = \max(0, T_{max1} * N_{match_k} - T_{QB2acc_{k-1}}) \tag{3.11}$$

減少する．

同様に，QB2 については，外部出力が発生した際に QB1 が空である場合，T_{max2} 間隔のパケット 1 つ分が欠落するため，QB2 の待機時間は

$$Dec_{out2} = \max(0, T_{max2} * N_{out_k} - T_{QB1acc_{k-1}}) \tag{3.12}$$

減少する．

最後に，$T_{max1} > T_{max2}$ であることから，各 QB において入出力レートの差が生じる．この結果，パケット転送が進行するにつれて，QB1 では待機時間が増加し，QB2 では待機時間が減少する傾向を示す．ランク k における巡回パケット数 N_k より，入出力レートの差から最大 $(T_{max1} - T_{max2}) * N_k$ だけ QB1 は増加し，QB2 は減少する．ただし，QB2 が空である場合，QB1,QB2 ともに入出力間隔が T_{max1} 以上となり，入出力レートの差が生じないことから前ランクにおける累積待機時間 $T_{QB2acc_{k-1}}$ を上限とする．また，例外として，当該ランクにおけるコピー発生時には，コピーステージにて T_{max2} の転送が生じるため，この上限に含まれない．以上のことから，QB1 の入出力レートの差による待機時間時間の増加は，

$$\begin{aligned} Inc_{rate1} = \min(T_{QB2acc_{k-1}}, (T_{max1} - T_{max2}) * N_k) \\ + (T_{max1} - T_{max2}) * N_{copy_k} \end{aligned} \tag{3.13}$$

となり，QB2 の入出力レートの差による待機時間時間の減少は，

$$Dec_{rate2} = \min(T_{QB2acc_{k-1}}, (T_{max1} - T_{max2}) * N_k) \tag{3.14}$$

となる．

以上のような各ランクにおける QB 内待機時間の増減を逐次積算することで，各 QB の最大待機時間を特定する．ただし，巡回パケット数が $\frac{\sum T_{f_i}}{T_{max}}$ 以下である場合，DDP 本来の緩衝能力に緩衝され，渋滞列が生じないことからいずれの QB にも待ち時間が生じない．よってその点を考慮し，ランク k における巡回パケット数が $\frac{\sum T_{f_i}}{T_{max}}$ 以下である場合，ランク k の QB1 における累積待機時間 T_{QB1acc_k} は，$N_k - \frac{\sum T_{f_i}}{T_{max}}$ で初期化する．さらに，求められた累積待機時間 T_{QB1acc_k}，T_{QB2acc_k} は当該ランク実行後の累積値を示すものであり，必ずしもランク内で発生し得る待機時間増加の最大値を表すものではない．そのため，本研究では最悪条件を想定し，当該ランクにおける実質的な累積待機時間は，各 QB における主要な待機時間増加要因のみを，前ランクにおける累積待機時間に加算することで算出する．以上の手順により導出した最大累積待機時間をそれぞれの QB の出力間隔で正規化を行い，最大 QB 蓄積パケット数を算出し，十分条件として採用する．

3.5　実行性能の見積りモデル

DDP によるプログラムのストリーム実行では，一定間隔で次世代のパケットが入力されるため，現在のプログラム実行が完了する前に次世代の実行が開始された場合には，巡回パケット数が世代を重ねるごとに増え続け，最終的にデッドロックが発生する恐れがある．よって，DDP が満たすべきプログラム実行性能は 1 世代分のプログラム実行時間がシステム要件で定義されたプログラム実行間隔よりも短くなることである．

DDP におけるプログラム実行はパケットがパイプライン上を 1 周するごとに 1 ランク分の処理が実行されるため，実行プログラムのランク数を R，1 周にかかる時間を $\sum T_{f_i}$ とすると，$R * \sum T_{f_i}$ で実行が完了する．ただし，ランク k の巡回パケット数 N_k が $\frac{\sum T_{f_i}}{T_{max}}$ を超えるとパケット転送の渋滞が発生し，1 周に要する時間が $\frac{\sum T_{f_i}}{T_{max}}$ を超えたパケット一つにつき T_{max} 増加する．また，巡回パケット数の増減が生じると，そのランクで巡回するパケッ

トの最後尾がずれる．以上の点を考慮して，プログラム実行時間の見積り式を考案した．

$$\begin{aligned} T_{exec} = R * \sum T_{f_i} \\ + T_{max} * \sum_{k=0}^{R} \max(0, N_k - \frac{\sum T_{f_i}}{T_{max}}) \\ + T_{max} * \sum_{k=0}^{R} \max(0, N_{in_k} - N_{out_k} + N_{copy_k} - N_{match_k}) \end{aligned} \tag{3.15}$$

3.6　結言

本章では，負荷変動耐性を担保する条件について定式化を行うにあたっての方針を述べた．続いて，プログラム実行時の負荷変動状況について増加減少の要因を整理し，プログラム構造から簡易的に定量化を行った．そして，その結果に基づいて負荷変動耐性を担保するための条件となる必要十分な QB 容量とプログラム実行性能の見積りモデルを提案した．

本研究で提案したモデルは単一プログラムのストリーム実行を想定していたが，実用的には異なる複数のプログラムを同時並行で実行することが想定される．このような場合には，各プログラムに対して本提案モデルを適用して得られた結果を足し合わせることで，システム全体としての負荷変動耐性条件の見積りが可能であると考えられる．

第 4 章

実装・評価

4.1　緒言

本章では, 前章で提案した負荷変動耐性条件の見積りモデルについての評価として，QB を搭載した DDP を FPGA 上に実装し，負荷変動が激しいプログラムを動作させた場合の回路シミュレーション結果との比較評価を行った結果と考察を述べる．

4.2　評価方法

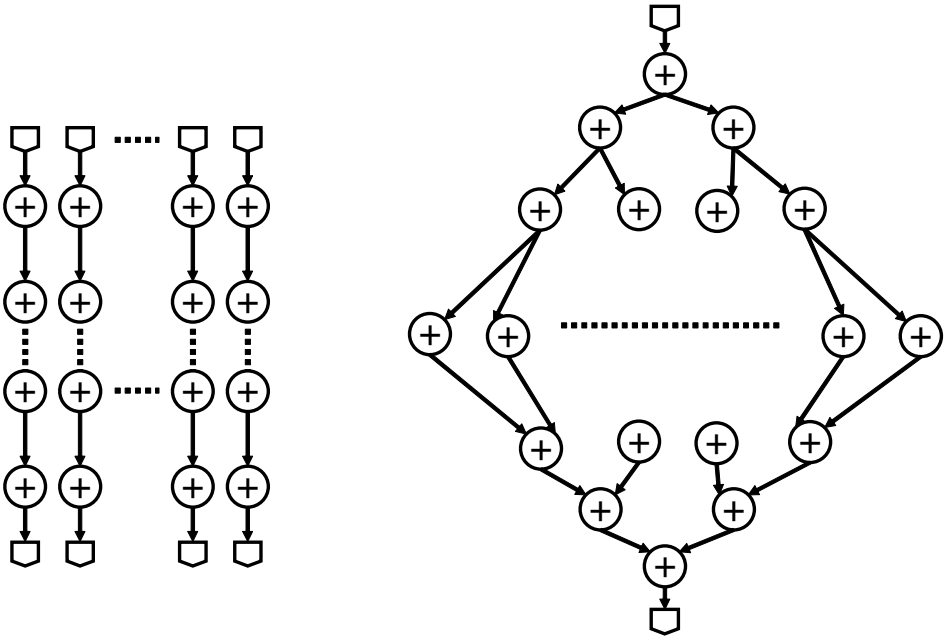
QB を適用した DDP を AMD 社製 FPGA チップ Zynq-7010 上に実装し，配置配線後の実遅延シミュレーションによる実行結果と提案モデルとの比較評価を行う．実装およびシミュレーションには，AMD 社製 FPGA 用 EDA ツール Vivado2023.1 を使用する．なおシミュレーションでは，各ステージの CP 信号の立ち上がり時刻を抽出し，時系列順にカウントすることで DDP 全体の巡回パケット数および各 QB に蓄積されたパケット数の算出を行う．またプログラムの実行時間は最初にパケットが入力されてから，最後の出力パケットが出力されるまでの時間を算出する．

4.2.1　評価用プログラムと実行条件

評価で用いるプログラムのデータフロー図を図 4.1，図 4.2，図 4.3 に示す．ここで実行するプログラムは負荷変動が大きいものとして，入力ソースを多数含むプログラムである P1，コピーを多数含むプログラムである P2，その両方を多数含むプログラムである P3 を

用いる．これらのプログラムは，QB の効果を顕著に示すため，本研究で扱う DDP のパイプライン段数である 8 を上回る最大負荷 (巡回パケット数) まで単調に増加し，その後単調に減少して巡回パケット数が 0 になるように設計した．また，現実的なプログラムとして 8PointFFT を P4 として用いる [9].

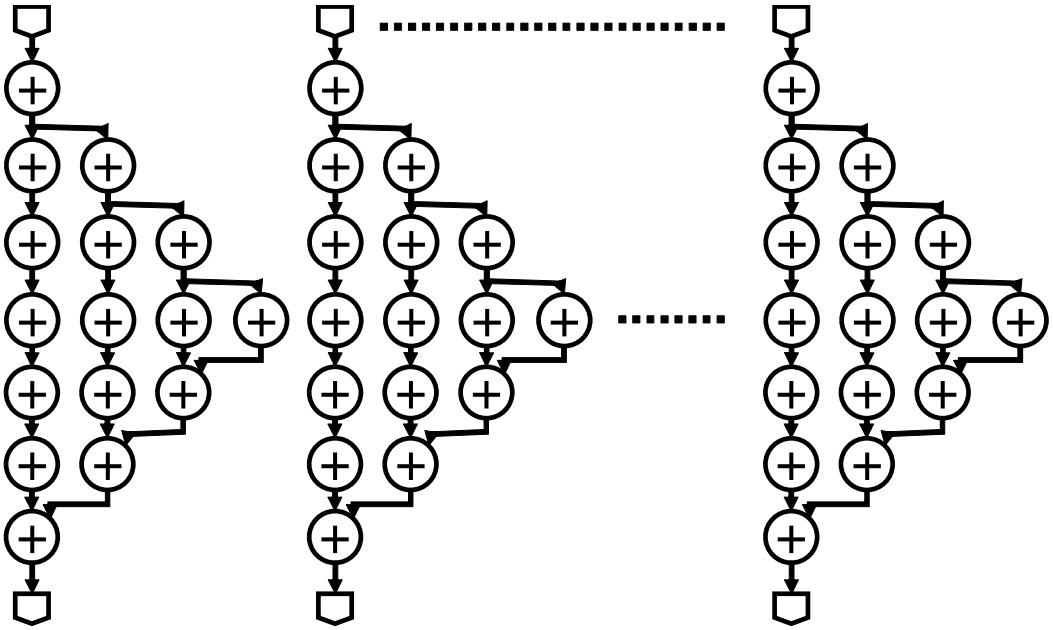
プログラムの実行条件として，入力ソースが複数ある場合には，外部からの入力パケットをパイプライン内部を周回してきたよりも優先して転送するものとし，DDP の最大入力間隔となる T_{max} で入力を行うものとする．さらに，アーキテクチャ関連のパラメタ値は，配置配線後の回路情報をもとに抽出された，$\sum T_{f_i} \approx 380[ns]$，$\sum T_{r_i} \approx 85[ns]$，$T_{max1} \approx 53[ns]$，$T_{max2} \approx 43[ns]$ として使用する．



$N = [16, 16, 16, 16, 16, 16, 16, 0]$
$N_{copy} = [0, 0, 0, 0, 0, 0, 0, 0]$
$N_{match} = [0, 0, 0, 0, 0, 0, 0, 0]$
$N_{in} = [16, 0, 0, 0, 0, 0, 0, 0]$
$N_{out} = [0, 0, 0, 0, 0, 0, 0, 16]$

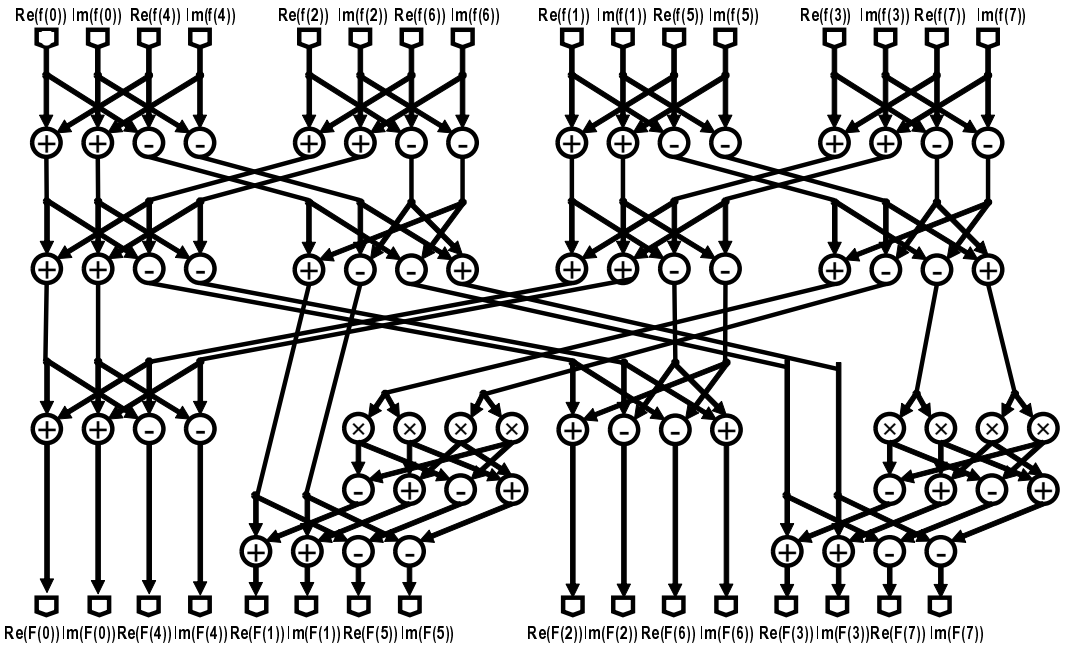
$N = [2, 4, 8, 16, 16, 8, 4, 2, 0]$
$N_{copy} = [1, 2, 4, 8, 0, 0, 0, 0, 0]$
$N_{match} = [0, 0, 0, 0, 0, 8, 4, 2, 1]$
$N_{in} = [1, 0, 0, 0, 0, 0, 0, 0, 0]$
$N_{out} = [0, 0, 0, 0, 0, 0, 0, 0, 1]$

図 4.1　P1(左図)，P2(右図) のプログラム構造



$$N = [32, 48, 64, 64, 48, 32, 0]$$
$$N_{copy} = [16, 16, 16, 0, 0, 0, 0]$$
$$N_{match} = [0, 0, 0, 0, 16, 16, 16]$$
$$N_{in} = [16, 0, 0, 0, 0, 0, 0]$$
$$N_{out} = [0, 0, 0, 0, 0, 0, 16]$$

図 4.2　P3 のプログラム構造



$$N = [32, 32, 16, 8, 0]$$
$$N_{copy} = [16, 16, 8, 0, 0]$$
$$N_{match} = [16, 16, 12, 8, 0]$$
$$N_{in} = [32, 0, 0, 0, 0]$$
$$N_{out} = [0, 0, 8, 0, 8]$$

図 4.3　P4(8 Point FFT) のプログラム構造

4.3　評価結果

前節の条件のもと，各プログラムをシミュレーション上で実行した際の負荷変動状況を図 4.4，図 4.5，図 4.6，図 4.7 に示す．ここで示される巡回パケット数は，各 QB 内で待機中のものを含めた DDP 全体を巡回するパケット数を時刻ごとに算出した値である．また，QB1，QB2 は各 QB 内に待機しているパケット数を時刻ごとに算出した値で，QB1+QB2 は同時刻上で各 QB で待機中のパケット数を合わせた値である．よって比較する値は，QB 容量の必要条件となる総 QB 容量は QB1+QB2 の最大値，QB 容量の十分条件となる各 QB 容量は，それぞれの最大値，そして実行時間は巡回パケット数が 1 になってから最終的に 0 になるまでの時間を用いる．

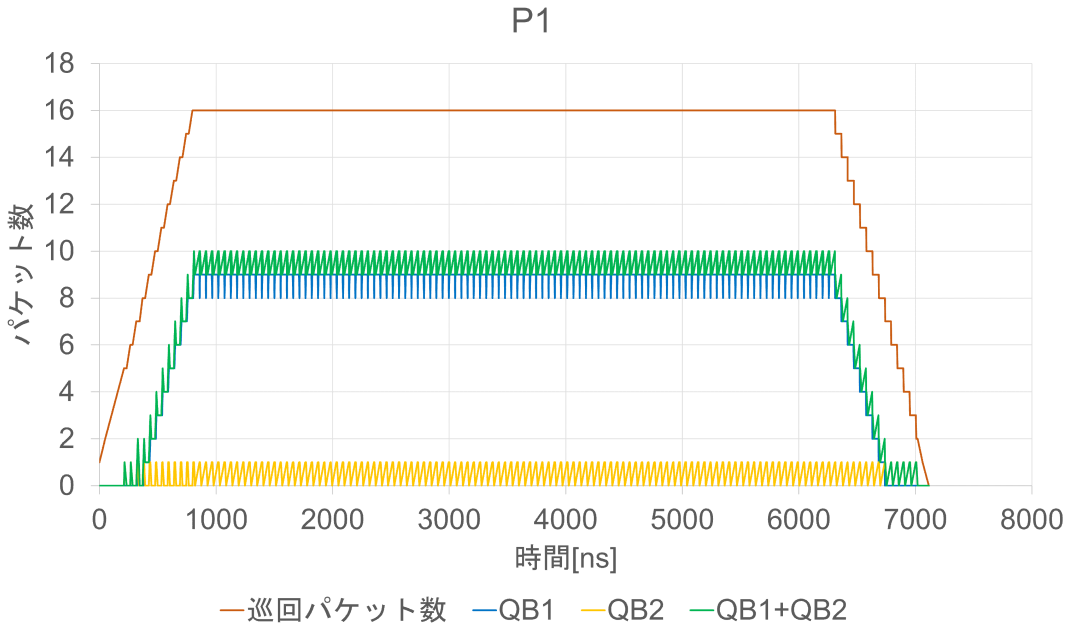


図 4.4　シミュレーション時の負荷変動状況 (P1)

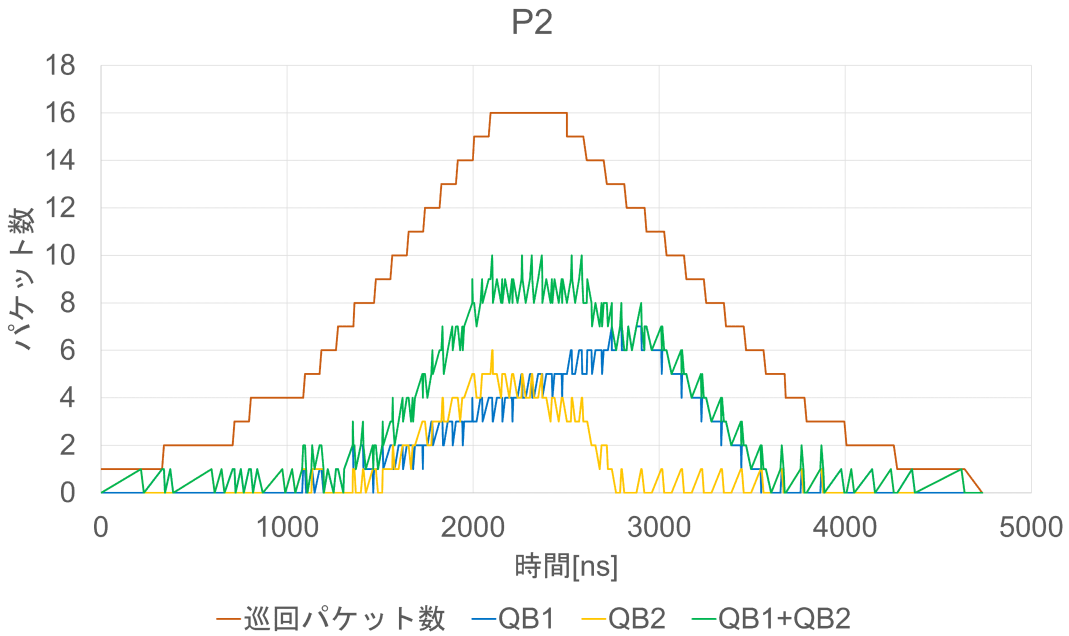


図 4.5　シミュレーション時の負荷変動状況 (P2)

4.3　評価結果

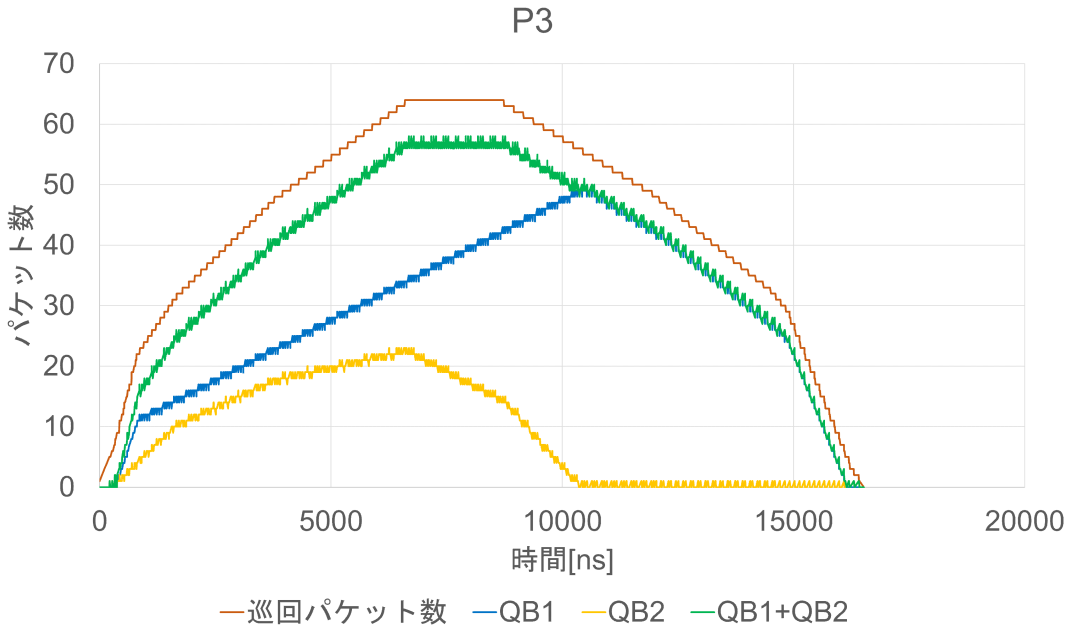


図 4.6　シミュレーション時の負荷変動状況 (P3)

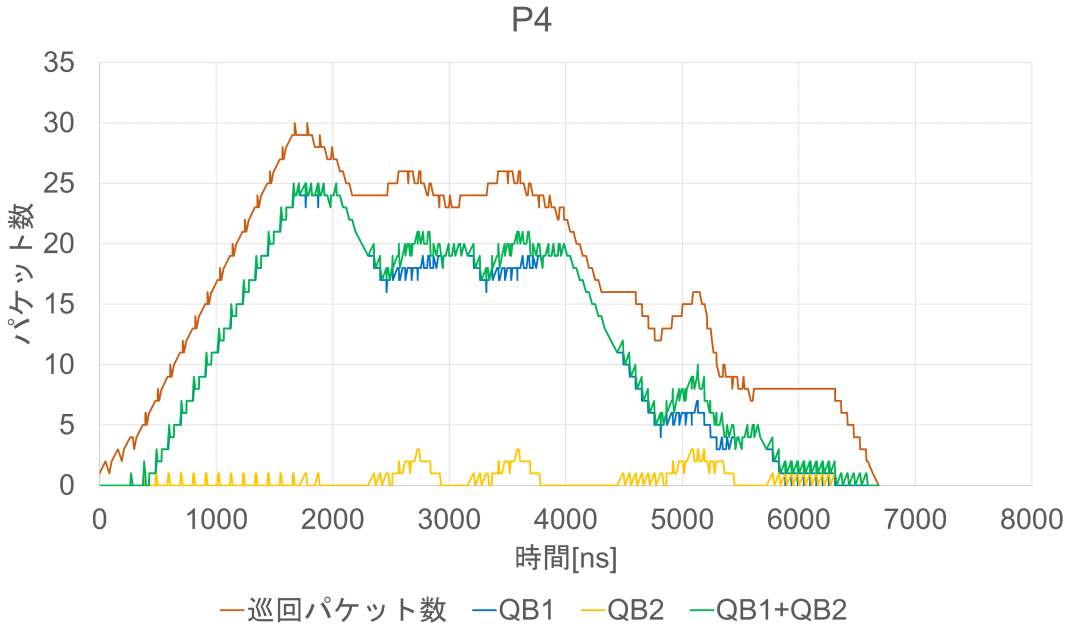


図 4.7　シミュレーション時の負荷変動状況 (P4)

次に，提案見積りモデルに基づく結果と回路シミュレーションによる検証結果を比較した結果を，表 4.1 に示す．検証の結果，必要条件となる総 QB 容量は，P1，P2，P3 において

完全に一致し，P4 においては誤差は 1 個で安全側に見積もられていることを確認した．

また，十分条件となる各 QB 容量については，P1 では QB1，QB2 ともに誤差 0 個，P2 では QB1 が誤差 1 個，QB2 が誤差 7 個，P3 では QB1 が誤差 1 個，QB2 が誤差 14 個，P4 では QB1 が誤差 0 個，QB2 が誤差 29 個となった．これらを合算した各 QB 容量の誤差は 0〜29 個の範囲に収まり，すべてのプログラムにおいて安全側に見積もられていることを確認した．

さらに，実行時間については，P1 で誤差 1.5%，P2 で誤差 12%，P3 で誤差 16%，P4 で誤差 1.5%となり，誤差は 1.5%〜16%の範囲で，いずれのプログラムにおいても安全側に見積もられていることが確認できた．

表 4.1　提案見積りモデルの評価結果

		P1	P2	P3	P4
提案	QB1+QB2	10	10	58	26
	QB1,QB2	9,1	8,13	51,37	25,32
	実行時間 [us]	7.22	5.32	19.19	6.79
検証	QB1+QB2	10	10	58	25
	QB1,QB2	9,1	7,6	50,23	25,3
	実行時間 [us]	7.11	4.74	16.52	6.69

4.4　考察

必要条件となる総 QB 容量の見積りモデルについて，P1〜P3 では回路シミュレーション結果と完全に一致し，P4 においても誤差 1 個で安全側に見積もられた．このことから，提案見積りモデルは総 QB 容量を高い精度で推定可能であり，必要条件の評価において高い妥当性を有していると考えられる．

一方で，十分条件となる各 QB 容量については，プログラムによって最大 29 個の誤差が生じており，必要条件と比較して誤差が大きくなる傾向が見られた．これは，プログラム実

行時における負荷変動状況を簡易的にモデル化したことにより，見積り誤差が生じた結果であると考えられる．特に本提案モデルでは，ランクごとに抽象化した負荷変動状況でモデル化を行っていることからランク内部での負荷変動を考慮できていない．そのため，今後さらなる見積り精度の向上を図るためには，各ランク内の詳細な挙動を反映した補正モデルの導入が必要である．ただし，いずれのケースにおいても誤差は安全側に収まっており，設計段階におけるリソース不足を回避するという観点からは，本提案モデルは有効であるといえる．

最後に，DDP が満たすべき実行性能の見積りモデルについては，P1 および P3 では比較的小さい誤差で推定されたのに対し，P2 および P4 では比較的大きな誤差が生じた．この原因は，本提案モデルにおいてパケット数の増減に伴う最後尾のずれを扱う項の仮定にあると考えられる．本提案モデルでは，ランクごとのパケット数の増減を算出する際，減少要因が多く負の値となる場合には，ずれが生じないものとして扱っていた．しかし，巡回パケット数が多く渋滞が発生している状況では，これらの減少要因が渋滞を緩和する方向に作用し，実際には最後尾が前方に移動する可能性がある．そのため，P2 や P4 のように高負荷状態でマッチングが多数発生するプログラムにおいて，実行性能の見積り誤差が大きくなったと考えられる．ただし，この場合においても，すべてのケースで見積り結果は安全側に収まっており，設計段階でのリソース不足を防ぐという観点から，本提案モデルは有効であるといえる．

4.5 結言

本章では, 提案した負荷変動耐性条件の見積りモデルについての評価として，QB を搭載した DDP を FPGA 上に実装し，負荷変動が激しいプログラムを動作させた場合の回路シミュレーション結果との比較評価を行った結果と考察を述べた．評価の結果，QB 容量の必要条件の見積りモデルについては高い精度で見積りが可能であることが示された．また，QB 容量の十分条件および実行時間については，簡易的にモデル化したことによる限界によ

りプログラム次第では大きな誤差が生じる結果となった．一方でいずれの状況においても安全側に誤差が生じていることからプログラムの特徴をランク毎に捉えることによって，各 QB 容量およびプログラム実行性能の上界を特定できる見通しが得られた．

第 5 章

結論

5.1 まとめ

近年，IoT(Internet of Things) 技術の普及に伴い，アプリケーションが高度化・多様化し，高性能かつ省電力なプロセッサが求められている．

この要件を満たす技術として，データ駆動型プロセッサ DDP(Data-Driven Processor) が注目されている．DDP はその動作原理からデータ依存関係にない処理を自然に並列実行可能で，さらにセルフタイム型パイプライン STP(Self-Timed Pipeline) 回路で実装することで省電力な動作が可能である．一方，DDP では周回パイプライン内を巡回するデータパケットの流れを円滑にしないとパイプライン処理性能を充分に発揮できず，最悪の場合，パケット転送が停止する可能性があるといった問題を内在している．この問題に対して，先行研究では，巡回パケット数の変動を吸収可能なキューバッファ機構 QB(Queue Buffer) を DDP パイプライン内に導入する方式が提案されている [4][5]．しかし，これら研究では，どの程度の容量の QB を導入すれば，どの程度の負荷変動耐性を備えられるのかは，十分には明らかにされていない．また，十分な QB 容量が設定されていたとしても，DDP 自体の処理性能を大幅に越える間隔でプログラムの実行が繰り返されると，QB に格納されるパケット数は際限なく増え続け，最終的に負荷変動を吸収しきれなくなる恐れがある．特に，DDP の実用が想定される IoT 用途においては，センサデータのフィルタリング処理のような一定間隔で入力されるデータの処理を行うストリーム実行が多いことから，次世代の入力までに処理を完了させるプログラム処理性能を有していることを保証する必要がある．

そこで，本研究では，応用プログラムの実行時の負荷変動状況に基づいて QB の適切な

容量およびプログラム実行性能を見積ることによって，負荷変動耐性を備えたデータ駆動型プロセッサを構成する方法を提案した．結果，プログラムの特徴をランク毎に捉えることによって，各 QB 容量およびプログラム実行性能について，誤差は生じるものの，上界を特定できる見通しが得られた．

5.2　今後の課題

- モデルの改良

  提案モデルの評価を行った結果，QB 容量の十分条件および実行性能の見積りモデルにおいて安全側ではあるもののプログラムによって大きな誤差が生じることが確認された．本モデルは既存手法に対してコスト削減を目的として，ランクごとの負荷変動という抽象化したモデルであることから，ある程度の誤差は生じうる．一方で誤差が大きくなればなるほど，より冗長なシステム構成になってしまうことから，可能な限り誤差を削減できるようにモデルの改善が求められる．

- 実機実装への適用

  本研究では，配置配線後の回路シミュレーションによる提案モデルの検証を行った．回路シミュレーションでは実装に用いられたすべての素子の遅延時間や配線遅延時間の情報を有しており，それらを基に実行されたプログラム実行状況の模擬は，実機上での実行に近しい形で実現される．一方，本研究で扱った DDP で用いられている C 素子は，所望の転送時間を確保するために複数の LUT をカスケード接続することで遅延時間を調整している．この構成では，各 LUT におけるシミュレーション結果と実機動作との間に生じる微小な誤差が段階的に累積する可能性がある．その結果，各ステージの転送制御遅延時間，さらには最終的なプログラム実行時間に無視できない差異が生じることが懸念される．以上より，今後は実機を用いた提案モデルの検証を実施するとともに，生じた誤差の補正手法を検討することが求められる．

- 定数演算バイパス型パイプラインへの拡張

5.2 今後の課題

本研究で扱った DDP アーキテクチャは，基本的構成要素 8 ステージを直列かつ環状に接続した基本的なパイプライン構造をとっていた．この構成は，構造が単純であるため，安定した動作が可能である一方で，二項演算のパケットと単項演算のパケットが同じパイプライン上を流れるため，MMCAM，MMRAM ステージでのマッチングがボトルネックとなっている．具体的に述べると，単項演算のパケットはマッチングを行わないため本来処理が不要であるはずの MMCAM，MMRAM ステージを通過する必要があり，パイプライン上を周回する時間が増加する．また，二項演算においては，待ち合わせが生じると，1 パケット分転送に空きが生じるため，実際の演算が行われる FP ステージのパケット流量が減少する．以上のことから，定数演算は MMCAM，MMRAM ステージを省略し，二項演算でできた転送の空きに対して補完するパイプライン構成が望ましいと考えられる．そのような機能を持ったパイプライン構成を図 5.1 に示す．この構成では，元の環状パイプライン構成に対して，定数演算を MMRAM ステージの後段にバイパスする B2 ステージ，MMRAM ステージから転送されるパケットと B2 ステージから転送されるパケットを調停する M2 ステージ，そして M2 ステージでパケットの衝突が発生した際に渋滞を発生させないため，B2 ステージ側に QB3 を追加が追加されている．この構成により，定数演算は周回時間を短縮でき、さらにマッチングで生じる転送の空きを QB3 で待機している定数演算で補完するため，処理性能の向上に寄与する可能性が高い．ただし，定数演算がプログラムのほとんどを占めるような応用プログラムでは，かえって周回時間を冗長にしてしまう恐れがあるため，今後は応用プログラムに応じた最適なパイプライン構成を選定する手法の検討が求められる．また，パイプライン構成が複雑化したことから，本研究で提案した負荷変動耐性条件の見積りモデルも直接適用することが出来ない．そのため，定数演算バイパス型パイプライン構成に対しても適用可能なモデルへの拡張が必要である．

5.2　今後の課題

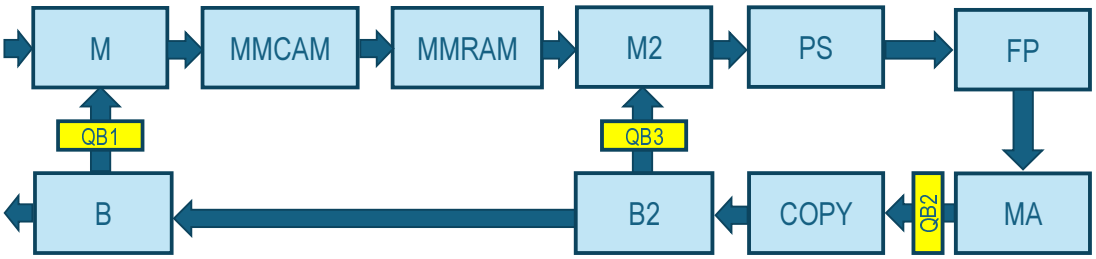


図 5.1　定数演算バイパス型パイプラインの構成

今後，以上の課題を解決することにより，応用システムの実装に要するハードウェアリソースや必要性能を簡易的かつ高精度に見積ることが可能となり，システム設計時の上流工程における早期見積りにより，設計コストの削減が可能になると考えられる．また，異なるパイプライン構成の DDP に対しても適用可能とすることにより，応用システムごとに最適なアーキテクチャを採択することが容易になると考えられる．

謝辞

本研究を進めるにあたり，様々なご指導を頂きました岩田誠教授には深く感謝申し上げます．ご多忙の中，研究内容の相談だけでなく，論文発表のスライド作成や練習，梗概や論文執筆時など様々な場面で助言して頂いたことで本研究を進めることが出来ました．

本研究の副査を引き受けてくださり，セミナー発表や修士論文発表において貴重なご意見を頂きました，横山和俊教授，松崎公紀教授には深く感謝申し上げます．

修士 1 年の頃から同研究室の先輩として，様々な面で支えてくださいました，Tamnuwat Valeeprakhon 氏，植元陸氏，坂口白磨氏，松坂拓海氏には深く感謝申し上げます．先輩方には，研究で行き詰まった際には真摯にご指導くださり，さらには就職活動に関するアドバイスや授業に関することなど様々なことを学ばせて頂きました．

同研究室の同輩として，ともに励ましあい努力してきた市ノ木一希氏，岡村健勝氏には深く感謝申し上げます．論文発表の練習や梗概，論文の校閲を相互に行い，高めあえたことで本研究を進めることが出来ました．

同研究室の後輩として，研究活動を支援して頂いた，大崎綾斗氏，門屋陽丈氏，苅田勇人氏，小松憲生氏，長崎雅季氏には深く感謝申し上げます．

最後に，生まれてから今日に至るまで，精神面や経済面での多大なご支援を頂きました，家族の方々に，心より感謝申し上げます．

以上，多大なるご支援ご協力を頂いた関係者の皆様には心よりお礼申し上げます．

参考文献

[1] 総務省, “令和 7 年版 情報通信白書 世界の IoT デバイス数の推移及び予測,” https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r07/html/datashu.html, 2026/2/20 閲覧

[2] NEC ソリューションイノベータ, “エッジコンピューティングとは？IoT の活用事例も解説,” https://www.nec-solutioninnovators.co.jp/sp/contents/column/20220225_edge-computing.html, 2026/2/20 閲覧

[3] H. Terada, S. Miyata, and M. Iwata, “DDMP ’s: Self-Timed Super-Pipelined Data-Driven Multimedia Processors,” Proceedings of the IEEE, vol. 87, no. 2, pp. 282-296, 1999.

[4] 上方輝彦, 岩田誠, 滝根哲哉, 寺田浩詔, “分散キューバッファを持つデータ駆動型プロセッサ Qv-x の性能評価,” 電気学会論文誌 C, Vol.116-C, No.11, pp.1295-1300, Nov.1996.

[5] 山下拓巳, “データ駆動型プロセッサ用セルフタイムパケットバッファ回路,” 高知工科大学学士学位論文, 2024.

[6] 高橋龍一, “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,” 高知工科大学学士学位論文, 2022.

[7] 三宮秀次, 大森洋一, 酒居敬一, 岩田誠, “自己タイミング型パイプラインシステムの性能見積りモデル,” 電子情報通信学会論文誌 A, Vol.J92-A, No.7, pp.477-486, 2009.

[8] Z. Hao, L. Liu and B. Tian, “The Principle and Applications of Asynchronous FIFO,” 2023 IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), Changchun, China, 2023, pp. 277-279

[9] 妻鳥恵三, “プロセッサ評価用 FPGA プラットフォーム上での FFT の実装・評価,” 高知工科大学学士学位論文, 2011.