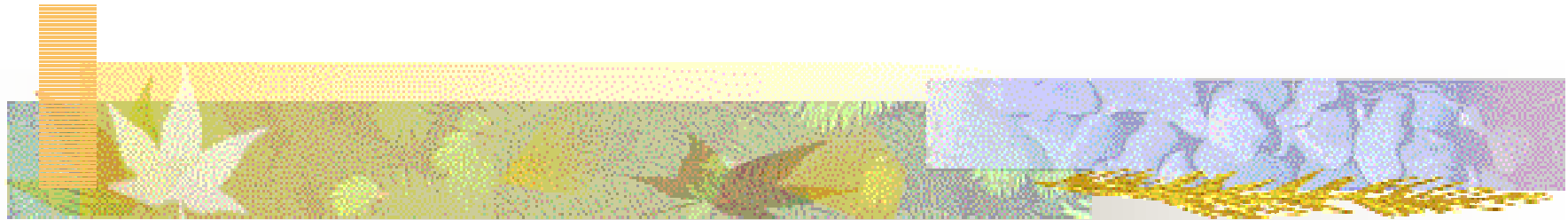


CPSY/DC研究会 (2003. 4. 18)

性能推定が容易な転送／演算分離型 並列画像処理システムの構成と評価



日立製作所日立研究所 吉田昌司
日立製作所日立研究所 池田光二
日立ハイテクノロジーズ 高根 淳

並列処理が実用的でない理由?

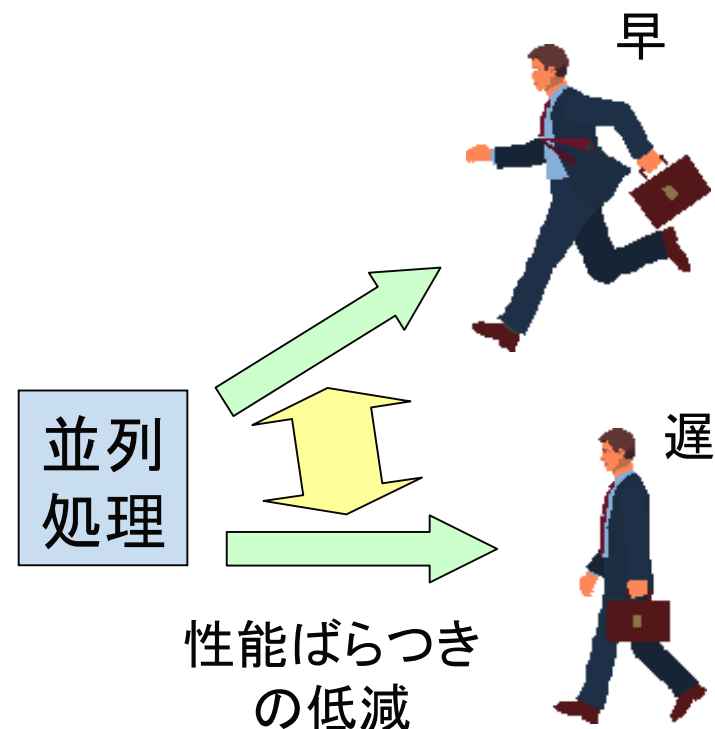
並列処理: 40年以上の長い歴史
⇔ 実用的に使われている例は少ない

- プログラムを書くのが難しい
- コストがかかる
- 並列化が有効な応用が少ない
- CPU n個で性能 n倍にならない

...

- 性能が良い場合と悪い場合
があって、実用に使いづらい

...

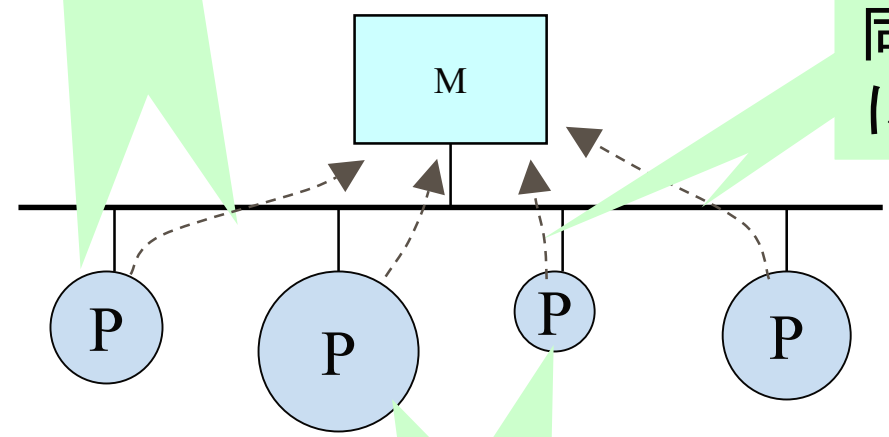


並列処理システムの性能ばらつきの原因

(1) 処理本体とオーバーヘッドを区別しにくい

(2) 複数プロセッサの同期や競合判定に時間がかかる

(3) 均等な負荷分散が難しい



並列処理システムの性能推定に関する 課題と解決策

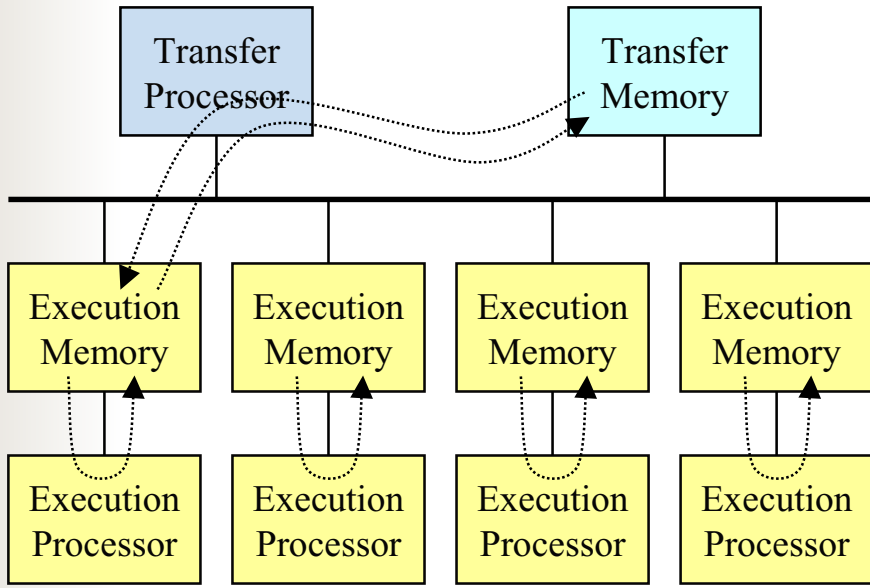
No	課題	解決策
1	処理本体とオーバーヘッドを区別しにくい	転送／演算分離型アーキテクチャ
2	複数プロセッサの同期に時間がかかる	フラグを用いた同期制御方式
3	均等な負荷分散が難しい	フィルタサイズを考慮した均等データ分割

第1の課題

No	課題	解決策
1	処理本体とオーバーヘッドを区別しにくい	転送／演算分離型アーキテクチャ
2	複数プロセッサの同期に時間がかかる	フラグを用いた同期制御方式
3	均等な負荷分散が難しい	フィルタサイズを考慮した均等データ分割

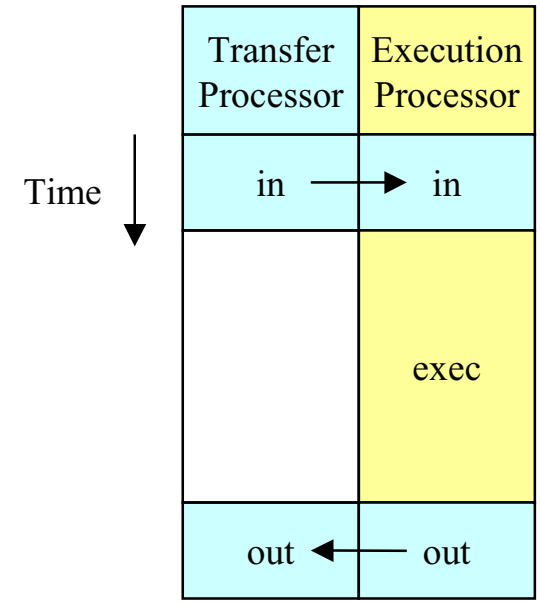
(1) 転送／演算分離型アーキテクチャ

転送プロセッサと演算プロセッサを分ける



- ・TP はTM, 全EMにアクセス可能
- ・EPは自分のEMのみアクセス可能

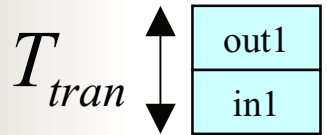
1行分処理手順



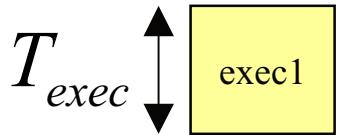
演算: 処理本体
 転送: オーバヘッド
 完全に分離

性能推定 (a) 転送ボトルネック

全体の処理時間は、
転送処理と演算処理の
バランスで決まる



1行分の
転送時間



1行分の
演算時間

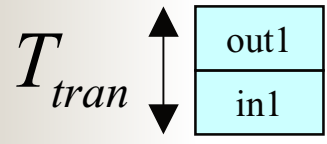
$T_{exec} < 3T_{tran}$ の場合

$$T_{system} = T_{tran}$$

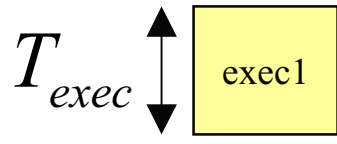
転送が支配的

Trans	Exec1	Exec2	Exec3	Exec4
out1 ← out1				
in1 → in1				
out2 ← out2				
in2 → in2				
out3 ← out3	exec1			
in3 → in3				
out4 ← out4		exec2		
in4 → in4				
out1 ← out1			exec3	
in1 → in1				
out2 ← out2				
in2 → in2				
out3 ← out3				
in3 → in3				
out4 ← out4				
in4 → in4				

性能推定 (b) 演算ボトルネック



1行分の
転送時間



1行分の
演算時間

$T_{exec} > 3T_{tran}$ の場合

$$T_{system} = \frac{T_{tran} + T_{exec}}{4}$$

演算が支配的

Trans	Exec1	Exec2	Exec3	Exec4
out1 ← out1				
in1 → in1				
out2 ← out2				
in2 → in2				
out3 ← out3				
in3 → in3				
out4 ← out4	exec1			
in4 → in4				
		exec2		
			exec3	
out1 ← out1				exec4
in1 → in1				
out2 ← out2				
in2 → in2				
out3 ← out3				
in3 → in3				
out4 ← out4				
in4 → in4				

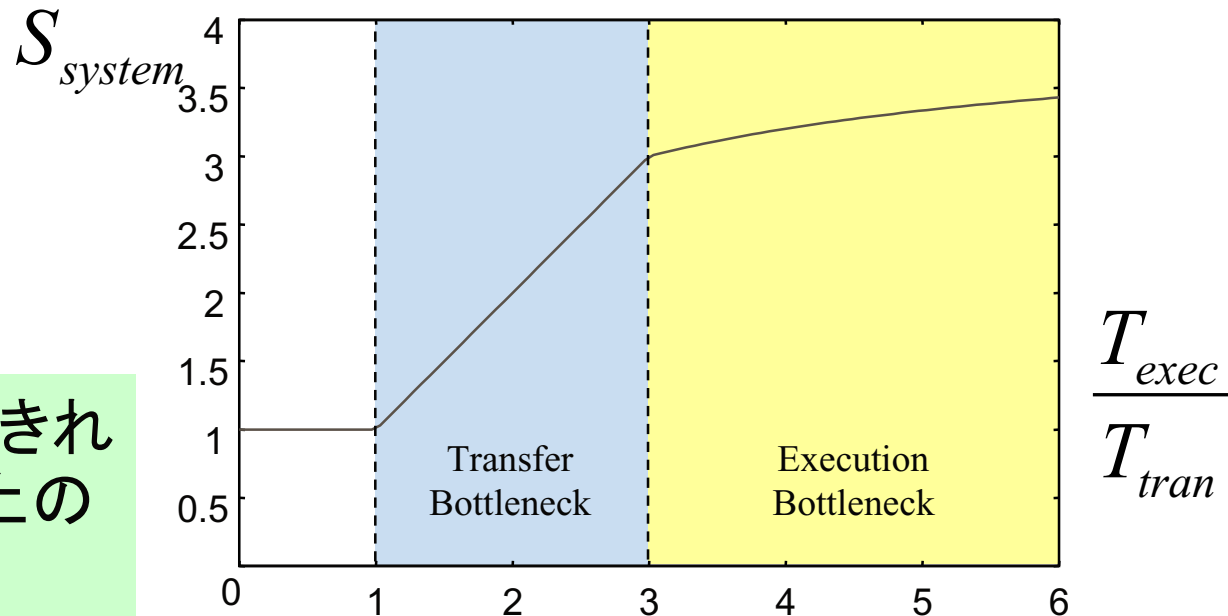
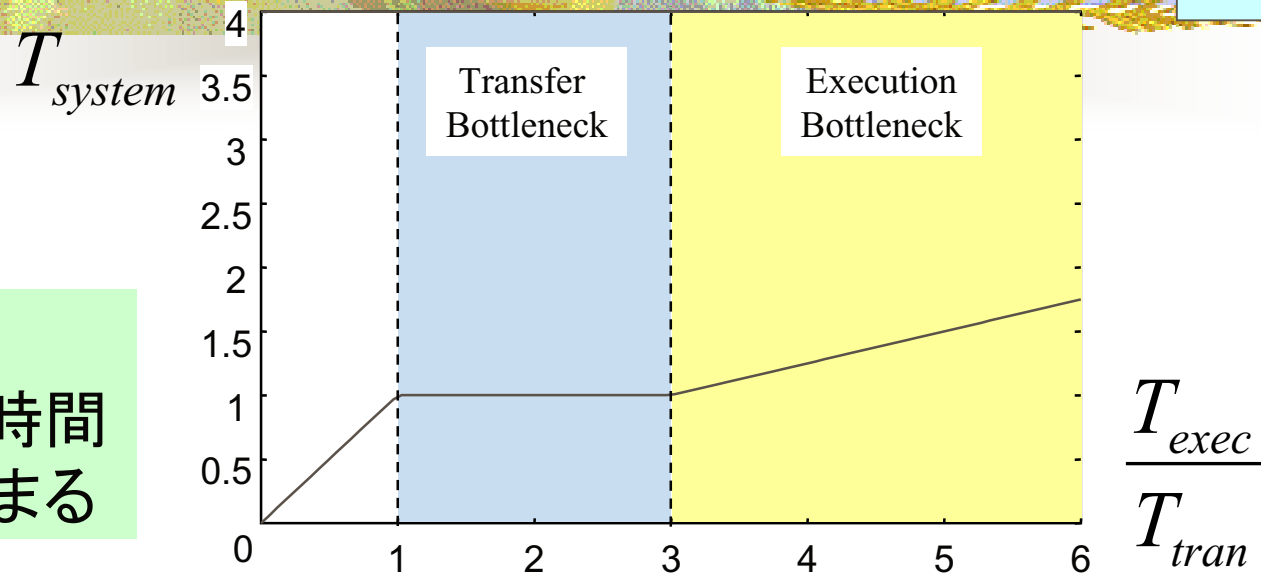
演算転送比と性能の関係

全体の性能は演算時間と転送時間の比で一意に決まる

$$S_{system} = \frac{T_1}{T_{system}}$$

単体CPUに対する速度向上比

演算ボトルネックにできれば、CPU3つ以上の性能が出る



複合関数合成による性能向上

関数 A

Trans	Exec1	Exec2	Exec3	Exec4
out1	out1			
in1	in1			
out2		out2		
in2		in2		
out3	A1		out3	
in3		A2	in3	
out4				out4
in4				in4
out1	← out1		A3	
in1				A4
out2	← out2	out2		
in2				
out3	← out3		out3	
in3				
out4	← out4			out4
in4				

関数 B

out1	out1			
in1	in1			
out2		out2		
in2		in2		
out3	B1		out3	
in3		B2	in3	
out4				out4
in4				in4
out1	← out1		B3	
in1				B4
out2	← out2	out2		
in2				
out3	← out3		out3	
in3				
out4	← out4			out4
in4				

関数A: 画像a + 画像b → 画像c

関数B: フィルタ (画像c) → 画像d

複合関数A+B

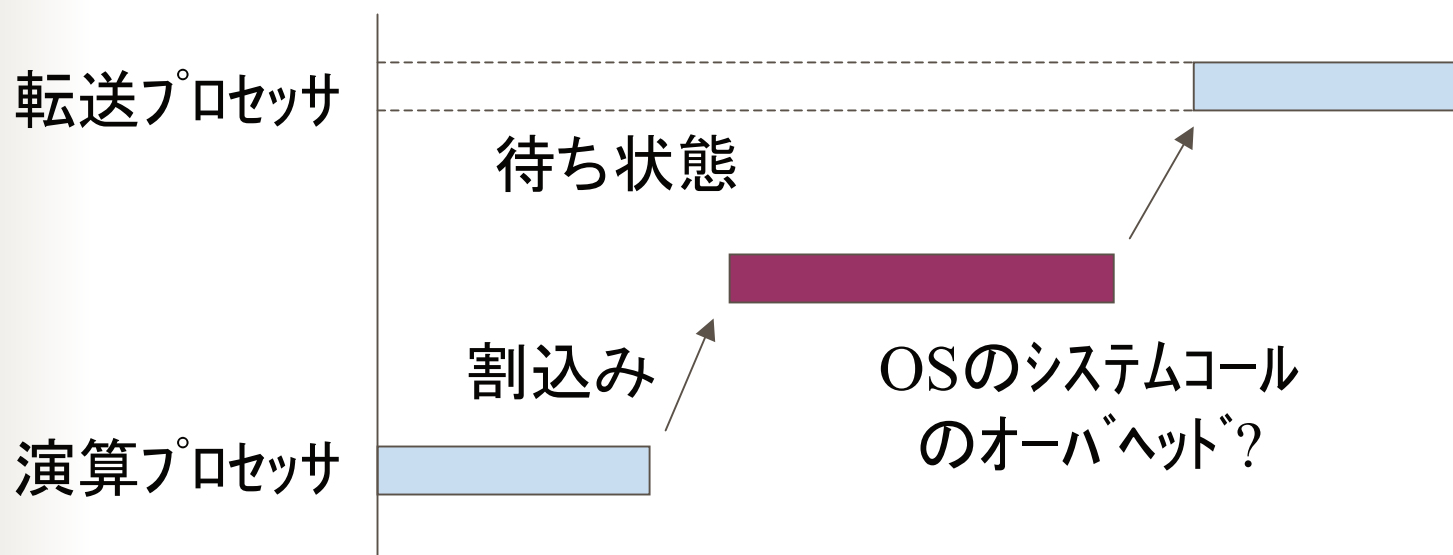
Trans	Exec1	Exec2	Exec3	Exec4
out1	out1			
in1	in1			
out2		out2		
in2		in2		
out3	A1		out3	
in3		A2	in3	
out4				out4
in4				in4
	B1		A3	
		B2		A4
out1	← out1		B3	
in1				B4
out2	← out2	out2		
in2				
out3	← out3		out3	
in3				
out4	← out4			out4
in4				

第2の課題

No	課題	解決策
1	処理本体とオーバーヘッドを区別しにくい	転送／演算分離型アーキテクチャ
2	複数プロセッサの同期に時間がかかる	フラグを用いた同期制御方式
3	均等な負荷分散が難しい	フィルタサイズを考慮した均等データ分割

割込みによる同期

並列処理システムでは
プロセス同士の同期や競合判定が必要



組込み向けOSではかなり考慮されているが、
それでも μ sオーダーかかり、その時間が伸び縮みする

(2) フラグを用いた同期制御方式

2ビットで同期制御できる

AR:アクセス権ビット

演算メモリのアクセス権

1: 演算プロセッサ

0: 転送プロセッサ

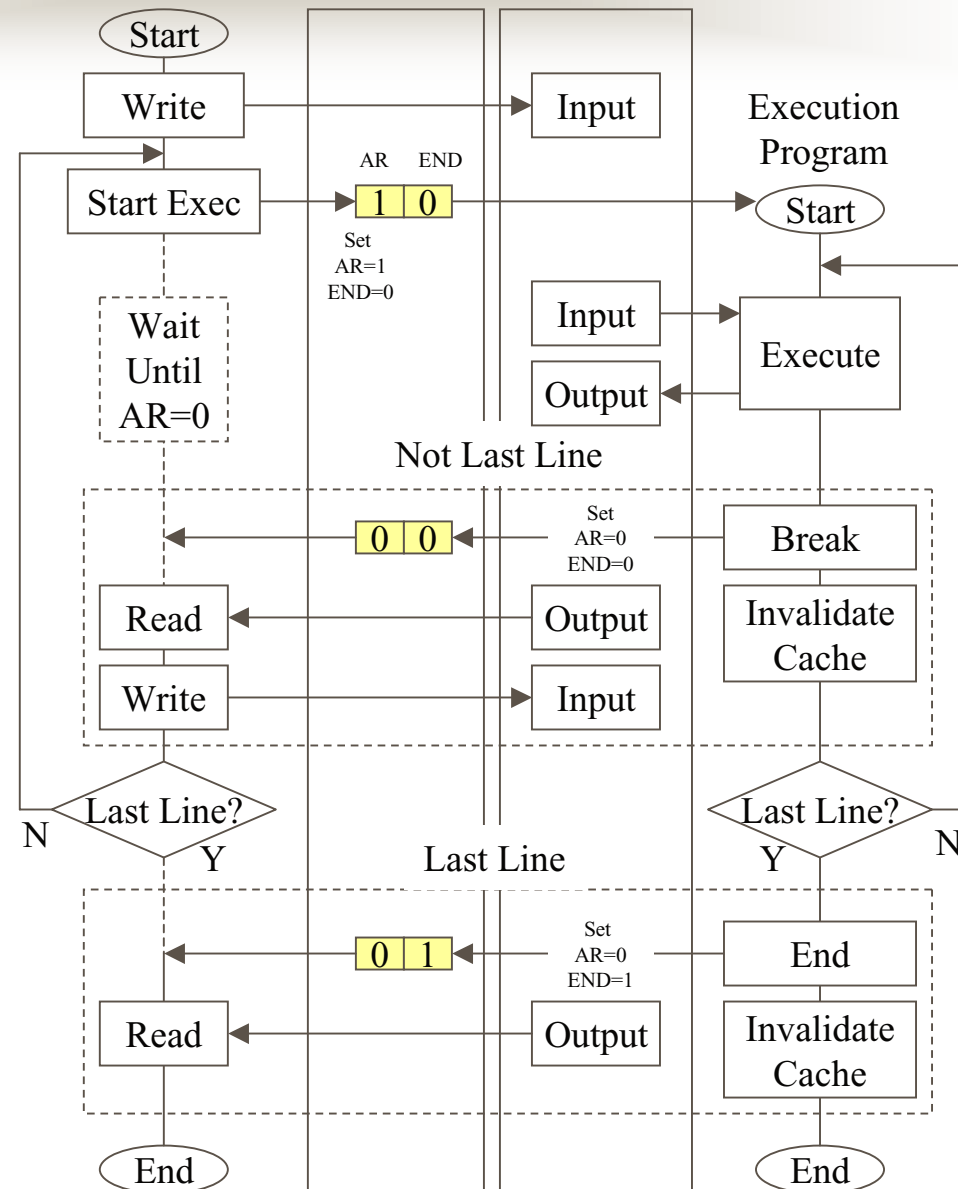
END:処理終了ビット

1: 処理終了

0: 処理継続

同期時間:数サイクル

性能ばらつきなし



第3の課題

No	課題	解決策
1	処理本体とオーバーヘッドを区別しにくい	転送／演算分離型アーキテクチャ
2	複数プロセッサの同期に時間がかかる	フラグを用いた同期制御方式
3	均等な負荷分散が難しい	フィルタサイズを考慮した均等データ分割

(3) 画像分割 (a) 横分割

画像処理ではフィルタ処理がよく用いられる

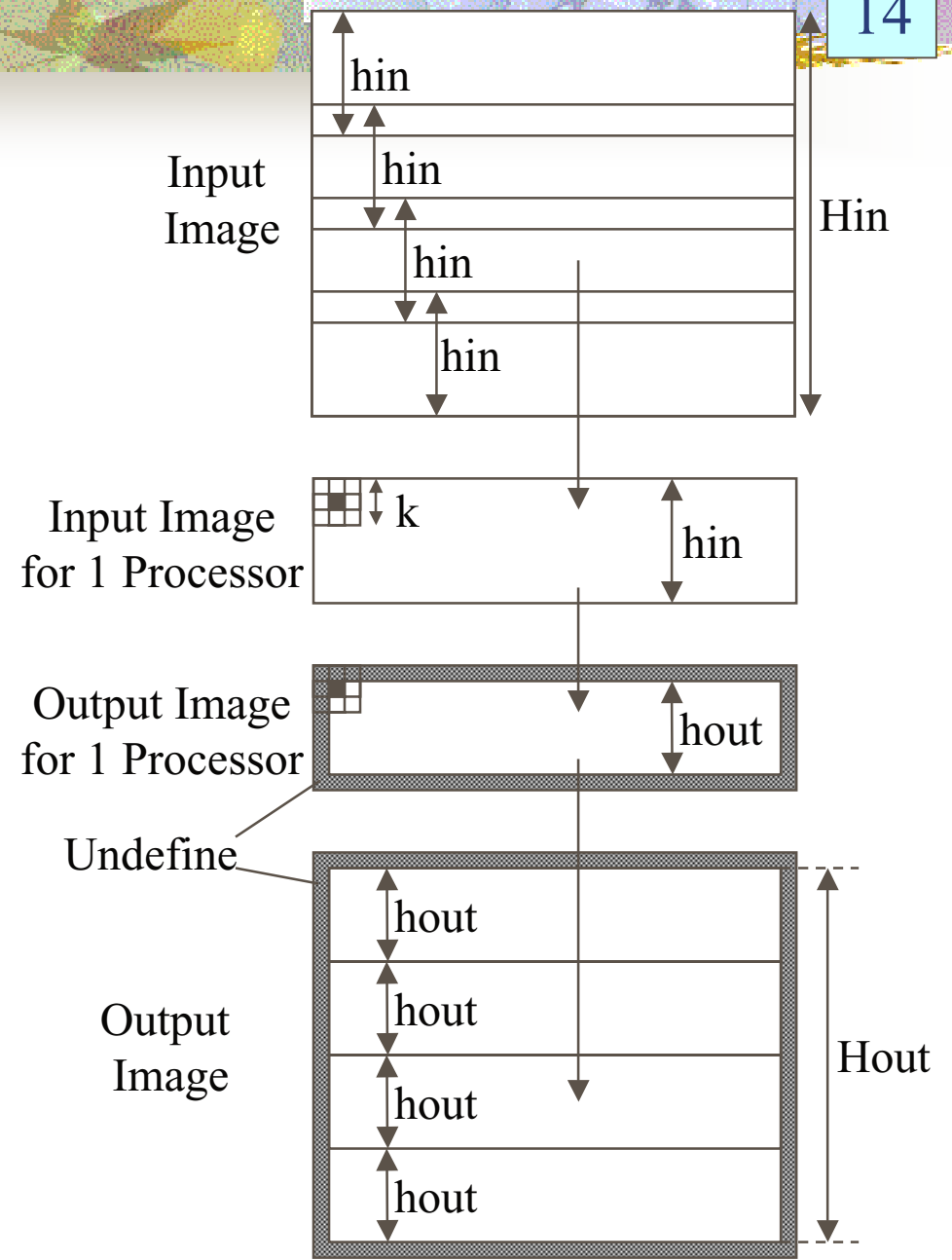
フィルタサイズ k を考慮した分割が必要

$$H_{out} = H_{in} - (k - 1)$$

$$h_{out} = \frac{H_{out}}{n} = \frac{H_{in} - (k - 1)}{n}$$

$$h_{in} = h_{out} + (k - 1)$$

$$= \frac{H_{in} + (n - 1)(k - 1)}{n}$$



(3) 画像分割 (b) 縦分割

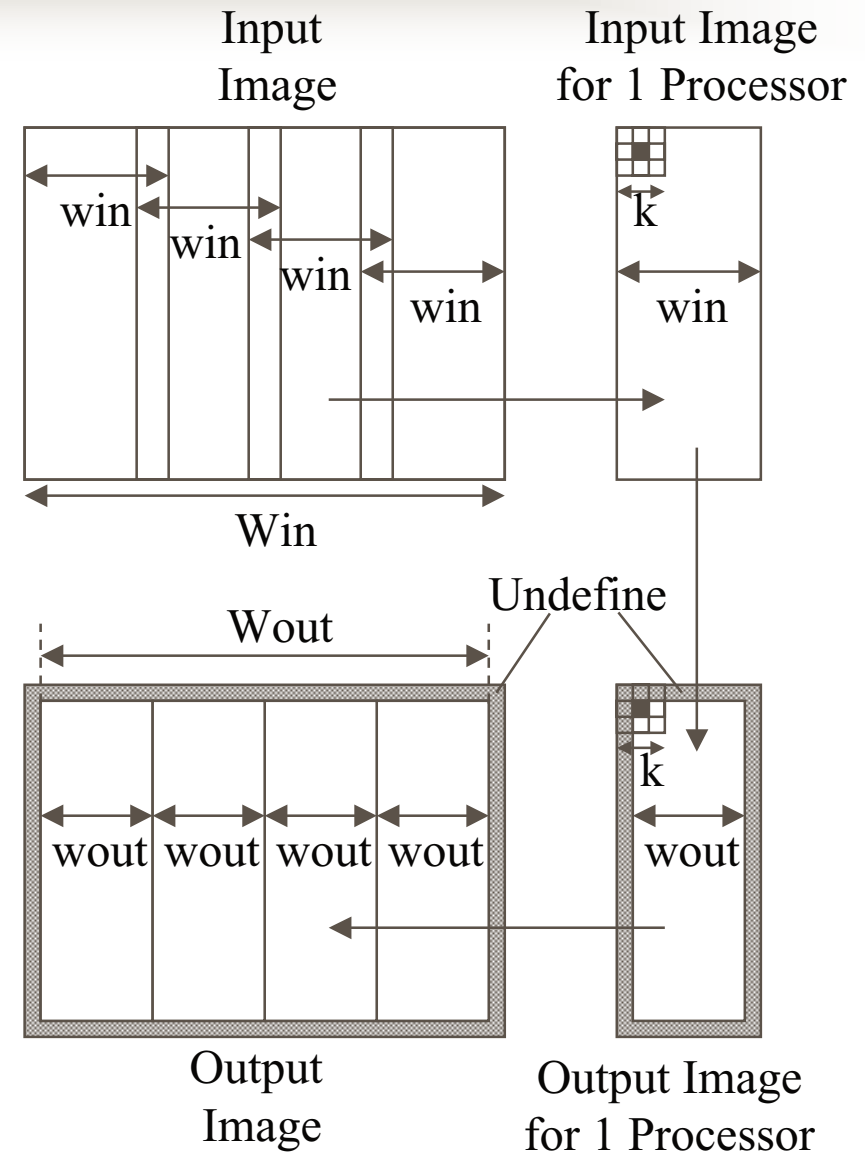
縦分割も同様に、
フィルタサイズ k を考慮した
分割が必要。

$$W_{out} = W_{in} - (k - 1)$$

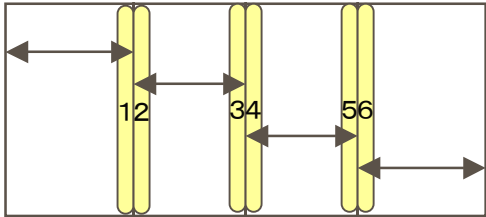
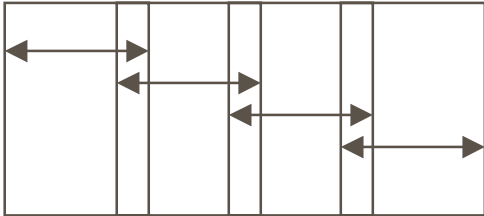
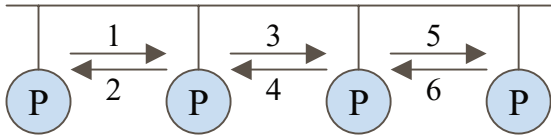
$$w_{out} = \frac{W_{out}}{n} = \frac{W_{in} - (k - 1)}{n}$$

$$w_{in} = w_{out} + (k - 1)$$

$$= \frac{W_{in} + (n - 1)(k - 1)}{n}$$



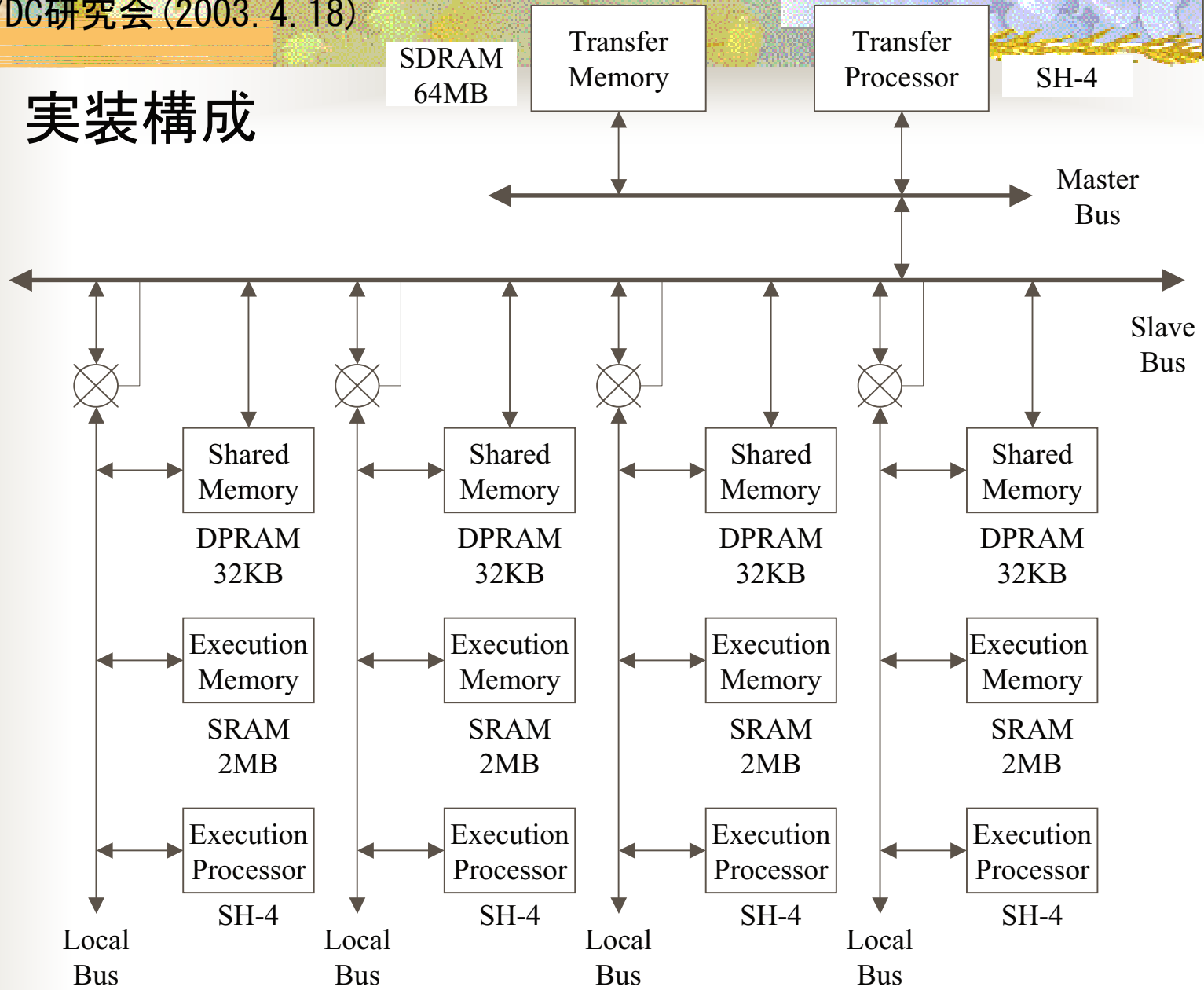
画像分割 割当て方式比較

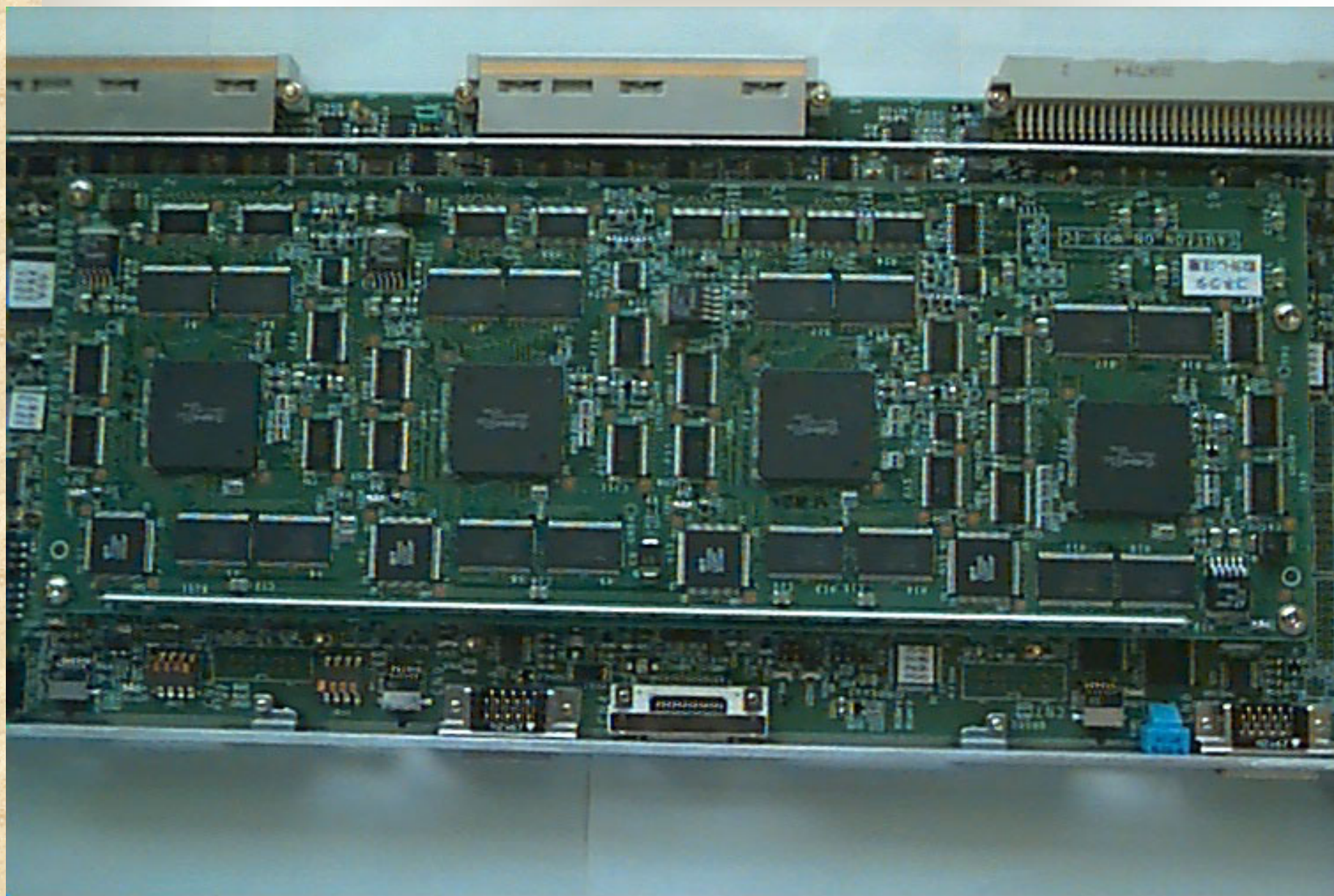
方式	(a) 重複を含めず分割	(b) 重複を含めて分割
構成		
転送P⇔演算P 転送	1画面分のみ →性能ばらつきなし W_{in}	重複分増加 →性能ばらつきなし $W_{in} + (n-1)(k-1)$
演算P間 転送	必要 →性能ばらつき発生 	不要 →性能ばらつきなし

実装と評価

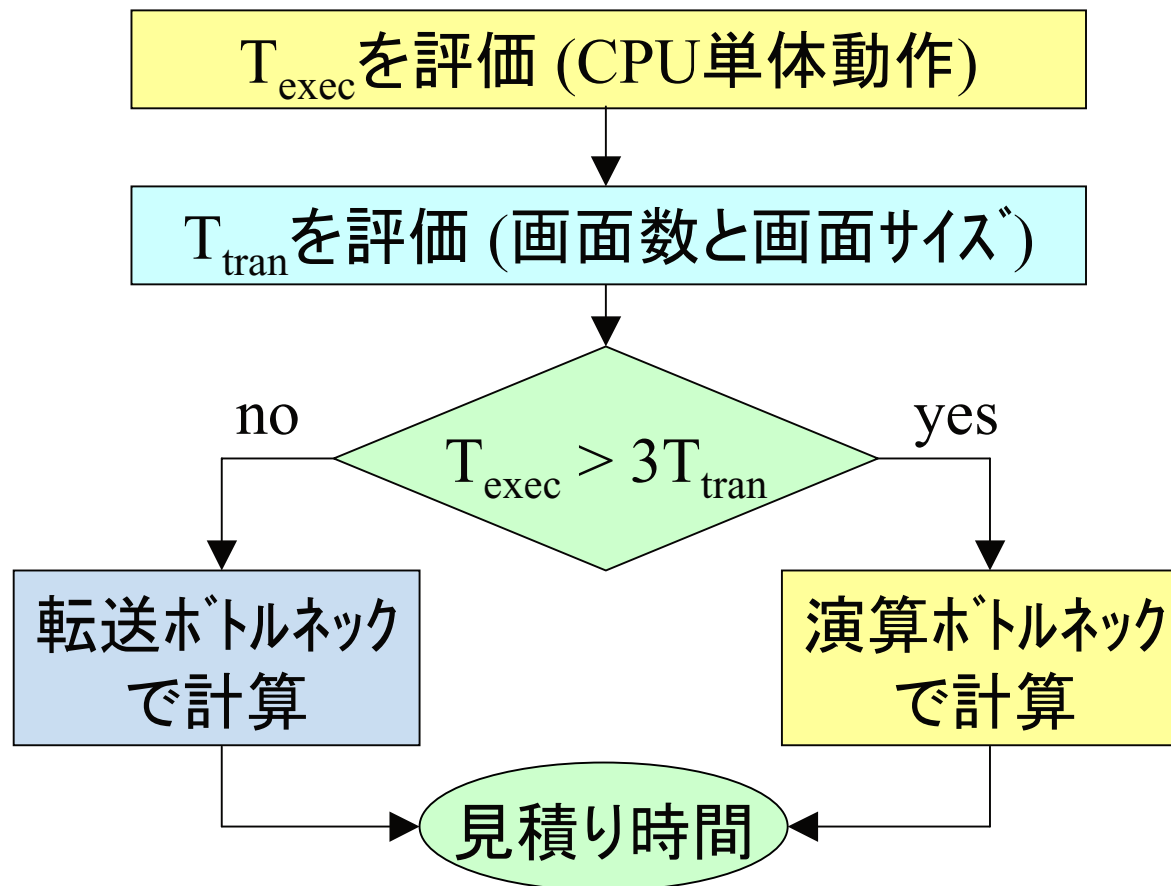
- 実装構成
- ボード写真
- 性能評価の手順
- 見積り時間と実測時間の比較

実装構成





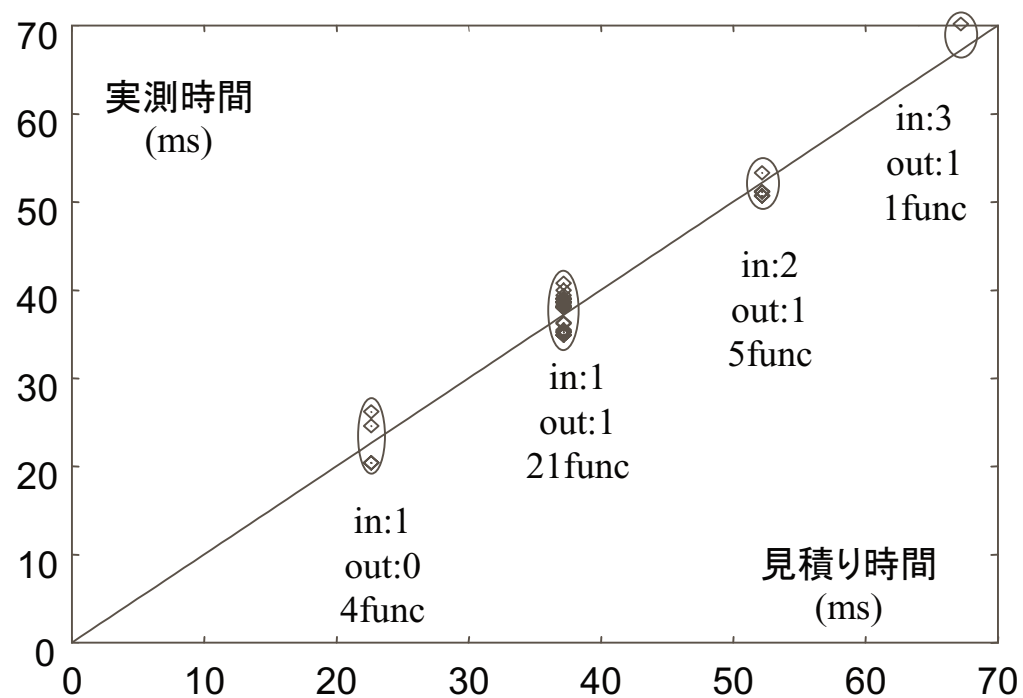
性能評価の手順



見積り時間と 実測時間の比較

画面サイズ: 512 × 512

関数型	関数の数	見積り 時間	誤差時間	
			平均	最大
1入力0出力	4	23ms	2.5ms	3.2ms
1入力1出力	21	38ms	1.4ms	3.1ms
2入力1出力	5	53ms	1.7ms	2.3ms
3入力1出力	1	68ms	2.0ms	2.0ms
計	31	-	1.6ms	3.2ms



結論

- (1) 性能をばらつかせる要素を低減させる工夫として以下を提案した。
 - 転送と演算を分離した並列画像処理アーキテクチャ
 - フラグを用いた同期制御方式
 - フィルタサイズを考慮した均等データ分割
- (2) 以上を実機に実装し、見積り誤差時間が平均1.6ms、最大3.2msであることを確認した。