

# Cerebras CS-2 上のプログラミングを支援する並列スケルトン

明治大学理工学部 若杉直椰 岩崎英哉

## 概要

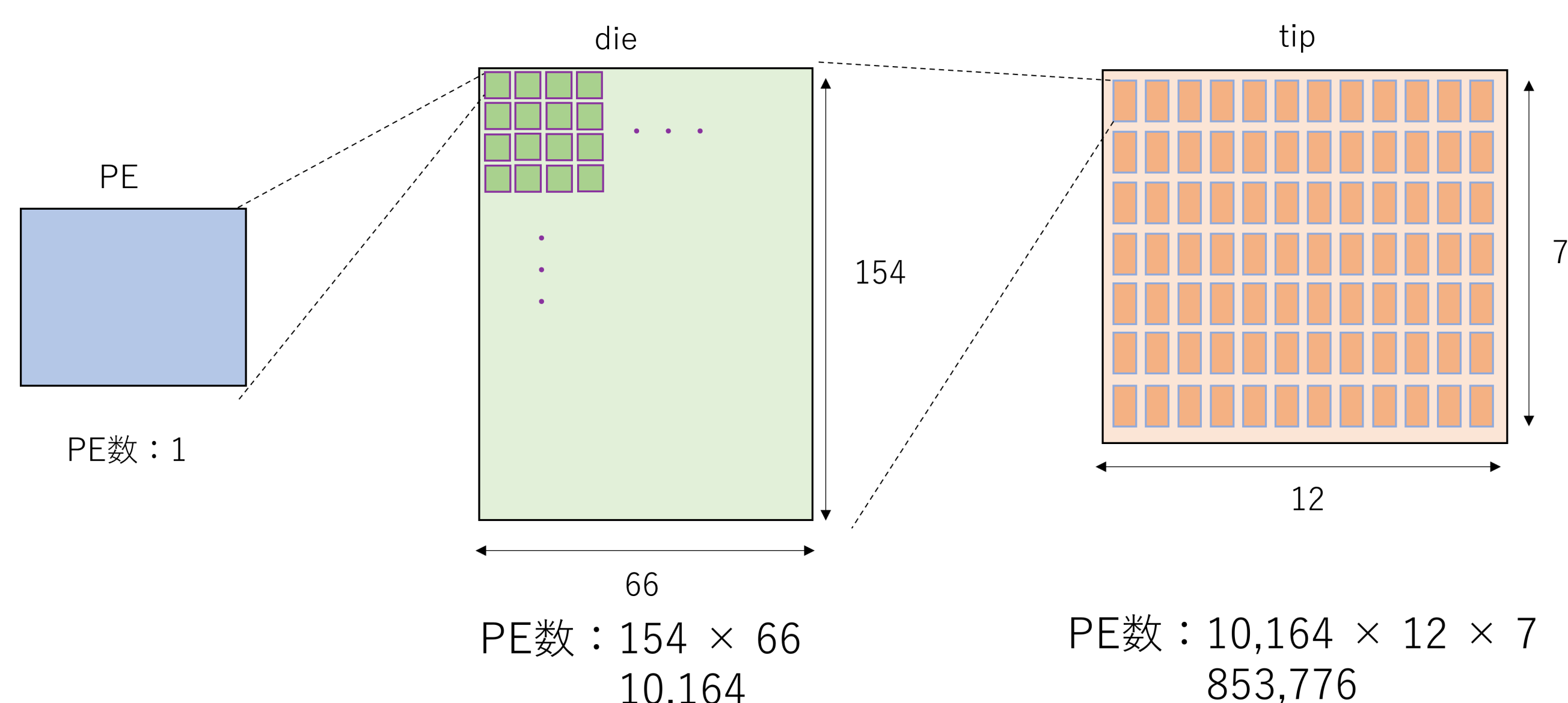
コーディング難易度の高い，機械学習用アクセラレータ Cerebras CS-2 のプログラム開発を支援する．

- ドメイン固有言語 (DSL) を導入し，並列スケルトンを用いて簡潔な記述を可能とした．
- DSLプログラムをCerebras CS-2上で動作するコードに変換する変換器を実装した．
- いくつかのプログラムを用いて，有用性を評価した．

## Cerebras CS-2: アーキテクチャ

大型半導体ウェハスケールエンジン(WSE-2)を搭載した，分散メモリ型のコンピュータ．

- ダイを切り出すことなく1枚の巨大なチップとすることで，チップ内部で大規模な並列計算を可能にしている [1] ．
- 密結合なPE同士で通信が完結するため，低レイテンシ通信が可能である．隣接PEへの通信は1サイクルで可能．



## Cerebras CS-2: プログラム開発

以下の三つのコードを書く必要がある

- ホストコード (Python): ホストとデバイス (CS-2) の間のデータの授受，CS-2上での関数実行を命令する．
- レイアウト (CSL): 使用するPEのレイアウトや，データの通信経路を決める．
- PEプログラム (CSL): 具体的なPE内での処理を記述する．

CSLはCS-2上のプログラムを記述する独自の構文を持つ言語であり，煩雑でハードルが高い．

## 提案手法1: 並列スケルトンの提供

並列スケルトン [2] は，並列プログラミングにおける頻出パターンを抽象化してライブラリ関数としたもの．

⇒ ユーザは，並列スケルトンを組み合わせて，並列性を意識することなくプログラムを記述できる．

1次元データ用の並列スケルトンとして，以下を提供する．

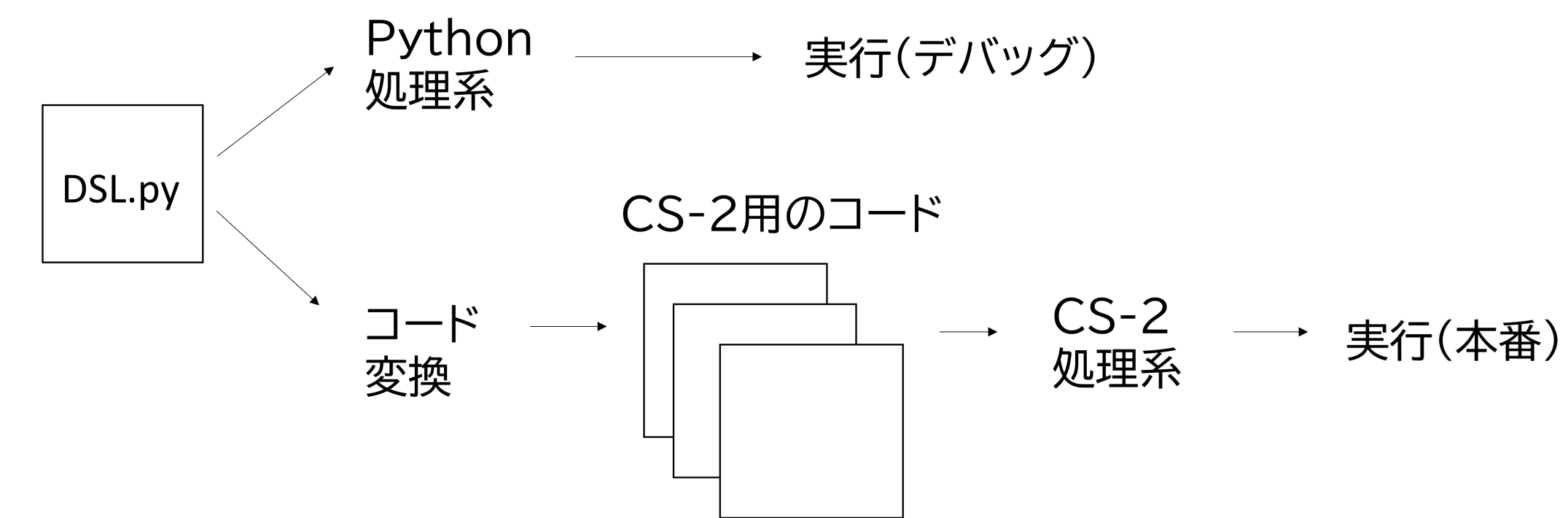
- map: リストに一様に関数適用したリストを返す．
- map\_ow: リストを直接書き換えるmap．
- zipwith: 2つのリストに一様に関数適用したリストを返す．
- zipwith\_ow: リストを直接書き換えるzipwith．
- reduce: 結合的な二項演算子を用いて要素を集積する．

2次元データについても同様のものを提供する．

## 提案手法2: DSLの導入

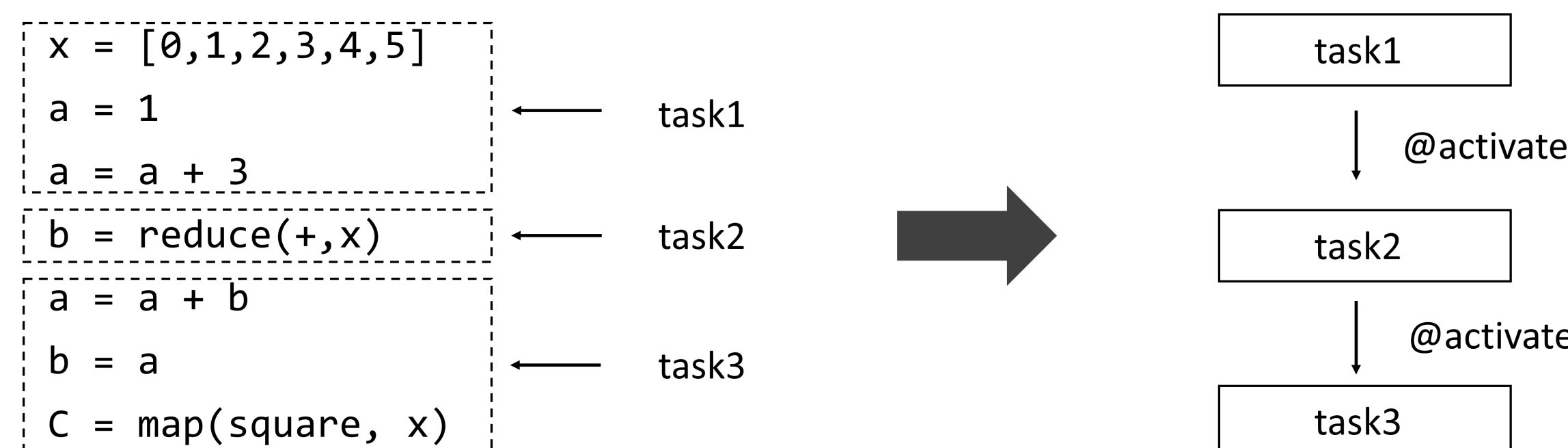
DSLをPythonの内部DSLとして定義する．

- CSLの煩雑な構文を回避する．
- Pythonの処理系による動作確認・デバッグを可能とする．



## コード変換器の実装

CSLはタスクに基づく継続渡し形式でコードを記述する．  
⇒ コード変換器は，DSLプログラムの制御フローを解析し，コードブロックをタスクへ割り当てる．



## 性能評価

提案したDSLによって，コードを簡潔に書くことができた．

```
1 param mempy_params: compile_struct;
2 param csl_params: compile_struct;
3 param data_per_pe: int;
4 param data_per_pe: int;
5
6 const mempy = @import_module("mempy/mempy", mempy_params);
7 const mempy = @import_module("mempy/mempy", mempy_params);
8 const mempy = @import_module("mempy/mempy", mempy_params);
9
10 var data_per_pe: int;
11 var result: [data_per_pe] f32;
12
13 var data_per: [f32] = @data;
14 var result: [f32] = @result;
15
16 var mempy = @mempy([data_per, result]);
17 var mempy = @mempy([data_per, result]);
18
19 var data_per = @get_data_per_pe;
20 var mempy = @mempy([data_per, result]);
21
22 var mempy = @mempy([data_per, result]);
23 var mempy = @mempy([data_per, result]);
24
25 var mempy = @mempy([data_per, result]);
26 var mempy = @mempy([data_per, result]);
27
28 const reduce_id: int;
29 const reduce_id: int;
30 const reduce_id: int;
```

PE プログラム (CSL)

```
1 param mempy_params: compile_struct;
2 param csl_params: compile_struct;
3 param data_per_pe: int;
4 param data_per_pe: int;
5
6 const mempy = @import_module("mempy/mempy", mempy_params);
7 const mempy = @import_module("mempy/mempy", mempy_params);
8 const mempy = @import_module("mempy/mempy", mempy_params);
9
10 var data_per_pe: int;
11 var result: [data_per_pe] f32;
12
13 var data_per: [f32] = @data;
14 var result: [f32] = @result;
15
16 var mempy = @mempy([data_per, result]);
17 var mempy = @mempy([data_per, result]);
18
19 var data_per = @get_data_per_pe;
20 var mempy = @mempy([data_per, result]);
21
22 var mempy = @mempy([data_per, result]);
23 var mempy = @mempy([data_per, result]);
24
25 var mempy = @mempy([data_per, result]);
26 var mempy = @mempy([data_per, result]);
27
28 const reduce_id: int;
29 const reduce_id: int;
30 const reduce_id: int;
```

レイアウト (CSL)

```
1 #!/usr/bin/env python
2 import argparse
3 import json
4 import sys
5
6 from cerebras_sdk.runtime import compile_struct
7 from cerebras_sdk.runtime import compile_struct
8 from cerebras_sdk.runtime import compile_struct
9
10 param = argparse.ArgumentParser()
11 param.add_argument("--name", help="the test name")
12 param.add_argument("--mempy", help="mempy for CS system")
13 args = param.parse_args()
14
15 with open(f"{args.name}.out.json", "w") as f:
16     compile_struct = compile_struct(args)
17     compile_struct = compile_struct(args)
18     compile_struct = compile_struct(args)
19     compile_struct = compile_struct(args)
20     compile_struct = compile_struct(args)
21     compile_struct = compile_struct(args)
22     compile_struct = compile_struct(args)
23     compile_struct = compile_struct(args)
24     compile_struct = compile_struct(args)
25     compile_struct = compile_struct(args)
26     compile_struct = compile_struct(args)
27     compile_struct = compile_struct(args)
28     compile_struct = compile_struct(args)
29     compile_struct = compile_struct(args)
30     compile_struct = compile_struct(args)
31     compile_struct = compile_struct(args)
32     compile_struct = compile_struct(args)
33     compile_struct = compile_struct(args)
34     compile_struct = compile_struct(args)
35     compile_struct = compile_struct(args)
36     compile_struct = compile_struct(args)
37     compile_struct = compile_struct(args)
38     compile_struct = compile_struct(args)
39     compile_struct = compile_struct(args)
40     compile_struct = compile_struct(args)
41     compile_struct = compile_struct(args)
42     compile_struct = compile_struct(args)
43     compile_struct = compile_struct(args)
44     compile_struct = compile_struct(args)
45     compile_struct = compile_struct(args)
46     compile_struct = compile_struct(args)
47     compile_struct = compile_struct(args)
48     compile_struct = compile_struct(args)
49     compile_struct = compile_struct(args)
50     compile_struct = compile_struct(args)
51     compile_struct = compile_struct(args)
52     compile_struct = compile_struct(args)
53     compile_struct = compile_struct(args)
54     compile_struct = compile_struct(args)
55     compile_struct = compile_struct(args)
56     compile_struct = compile_struct(args)
57     compile_struct = compile_struct(args)
58     compile_struct = compile_struct(args)
59     compile_struct = compile_struct(args)
60     compile_struct = compile_struct(args)
61     compile_struct = compile_struct(args)
62     compile_struct = compile_struct(args)
63     compile_struct = compile_struct(args)
64     compile_struct = compile_struct(args)
65     compile_struct = compile_struct(args)
66     compile_struct = compile_struct(args)
67     compile_struct = compile_struct(args)
68     compile_struct = compile_struct(args)
69     compile_struct = compile_struct(args)
70     compile_struct = compile_struct(args)
71     compile_struct = compile_struct(args)
72     compile_struct = compile_struct(args)
73     compile_struct = compile_struct(args)
74     compile_struct = compile_struct(args)
75     compile_struct = compile_struct(args)
76     compile_struct = compile_struct(args)
77     compile_struct = compile_struct(args)
78     compile_struct = compile_struct(args)
79     compile_struct = compile_struct(args)
80     compile_struct = compile_struct(args)
81     compile_struct = compile_struct(args)
82     compile_struct = compile_struct(args)
83     compile_struct = compile_struct(args)
84     compile_struct = compile_struct(args)
85     compile_struct = compile_struct(args)
86     compile_struct = compile_struct(args)
87     compile_struct = compile_struct(args)
88     compile_struct = compile_struct(args)
89     compile_struct = compile_struct(args)
90     compile_struct = compile_struct(args)
91     compile_struct = compile_struct(args)
92     compile_struct = compile_struct(args)
93     compile_struct = compile_struct(args)
94     compile_struct = compile_struct(args)
95     compile_struct = compile_struct(args)
96     compile_struct = compile_struct(args)
97     compile_struct = compile_struct(args)
98     compile_struct = compile_struct(args)
99     compile_struct = compile_struct(args)
100    compile_struct = compile_struct(args)
```

ホストコード (Python)

```
1 import skeleton
2
3 from dataclass import Data
4
5 data = Data()
6
7 def f(x):
8     return x + 1
9
10 def g(x):
11     return x * 2
12
13 data = Data()
14 data.data = [1, 2, 3, 4, 5, 6, 7, 8]
15
16 mempy = mempy(f, data)
17 mempy = mempy(f, data)
18 mempy = mempy(f, data)
19 mempy = mempy(f, data)
20 mempy = mempy(f, data)
21 mempy = mempy(f, data)
22 mempy = mempy(f, data)
23 mempy = mempy(f, data)
24 mempy = mempy(f, data)
25 mempy = mempy(f, data)
26 mempy = mempy(f, data)
27 mempy = mempy(f, data)
28 mempy = mempy(f, data)
29 mempy = mempy(f, data)
30 mempy = mempy(f, data)
31 mempy = mempy(f, data)
32 mempy = mempy(f, data)
33 mempy = mempy(f, data)
34 mempy = mempy(f, data)
35 mempy = mempy(f, data)
36 mempy = mempy(f, data)
37 mempy = mempy(f, data)
38 mempy = mempy(f, data)
39 mempy = mempy(f, data)
40 mempy = mempy(f, data)
41 mempy = mempy(f, data)
42 mempy = mempy(f, data)
43 mempy = mempy(f, data)
44 mempy = mempy(f, data)
45 mempy = mempy(f, data)
46 mempy = mempy(f, data)
47 mempy = mempy(f, data)
48 mempy = mempy(f, data)
49 mempy = mempy(f, data)
50 mempy = mempy(f, data)
51 mempy = mempy(f, data)
52 mempy = mempy(f, data)
53 mempy = mempy(f, data)
54 mempy = mempy(f, data)
55 mempy = mempy(f, data)
56 mempy = mempy(f, data)
57 mempy = mempy(f, data)
58 mempy = mempy(f, data)
59 mempy = mempy(f, data)
60 mempy = mempy(f, data)
61 mempy = mempy(f, data)
62 mempy = mempy(f, data)
63 mempy = mempy(f, data)
64 mempy = mempy(f, data)
65 mempy = mempy(f, data)
66 mempy = mempy(f, data)
67 mempy = mempy(f, data)
68 mempy = mempy(f, data)
69 mempy = mempy(f, data)
70 mempy = mempy(f, data)
71 mempy = mempy(f, data)
72 mempy = mempy(f, data)
73 mempy = mempy(f, data)
74 mempy = mempy(f, data)
75 mempy = mempy(f, data)
76 mempy = mempy(f, data)
77 mempy = mempy(f, data)
78 mempy = mempy(f, data)
79 mempy = mempy(f, data)
80 mempy = mempy(f, data)
81 mempy = mempy(f, data)
82 mempy = mempy(f, data)
83 mempy = mempy(f, data)
84 mempy = mempy(f, data)
85 mempy = mempy(f, data)
86 mempy = mempy(f, data)
87 mempy = mempy(f, data)
88 mempy = mempy(f, data)
89 mempy = mempy(f, data)
90 mempy = mempy(f, data)
91 mempy = mempy(f, data)
92 mempy = mempy(f, data)
93 mempy = mempy(f, data)
94 mempy = mempy(f, data)
95 mempy = mempy(f, data)
96 mempy = mempy(f, data)
97 mempy = mempy(f, data)
98 mempy = mempy(f, data)
99 mempy = mempy(f, data)
100 mempy = mempy(f, data)
```

DSL

3つのプログラムを対象に，コード長を比較した．

コードの種類	DSL	手書きのコード				DSL / 総和[%]
		code.csl	layout.csl	run.py	手書きコードの総和	
行列積	12	143	43	34	220	5.5
分散	16	158	43	34	235	6.8
モンテカルロ	17	146	44	41	231	7.4

コード長を約10%まで減少させることができた．

分散を求めるプログラムにおいて，シミュレータを用いて実行し，サイクル数を比較した．

コードの種類	サイクル数 [回]	割合
生成コード	7532	121
手書きのコード	6225	100

生成コードのサイクル数は約1.2倍となった．

## 今後の課題

- map, zipwithの融合等による，実行速度の向上
- 並列スケルトンの拡張

## 参考文献

- Takaaki Miyajima, Ryunosuke Matuzaki, Leon Fukuoka. "STREAM Benchmark on Cerebras WSE-2". ISC 2024. Poster.
- 岩崎英哉, 胡振江. "スケルトン並列プログラム". 情報処理. Vol46. No10 pp 1158-1162 (2005)