第27回プログラミングおよびプログラミング言語ワークショップ(PPL 2025) カテゴリ3

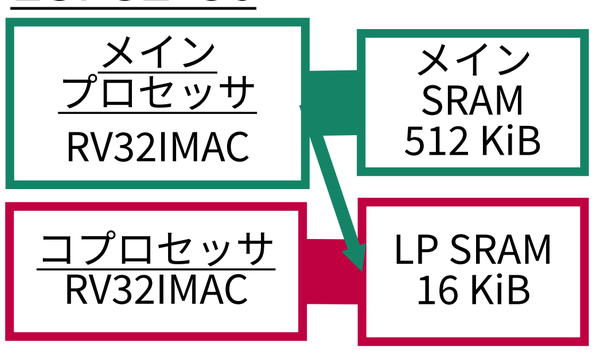
組込みデバイスのメモリ保護ユニットを活用したメモリ管理手法に向けた事前評価

鈴木豪，渡部卓雄，森口草介（東京科学大学情報理工学院）

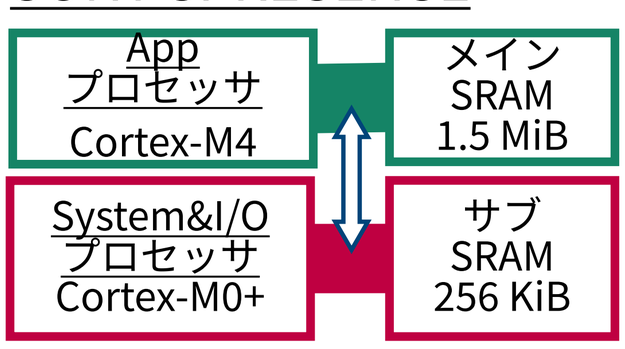
1.組込みデバイスにおける ヘテロジニアスメモリアーキテクチャ

複雑なトポロジで複数の演算装置や記憶装置が
接続されている組込みデバイス→実時間性や省電力性

ESP32-C6



SONY SPRESENSE



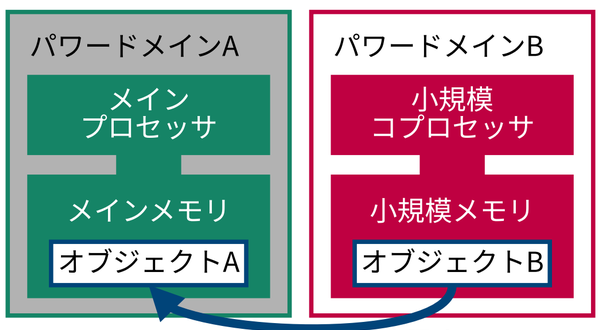
メモリ保護ユニット (MPU: Memory Protection Unit)
特定のメモリ領域へのアクセスを禁止できる
（Memory Management Unitのサブセット）

2.ユースケース

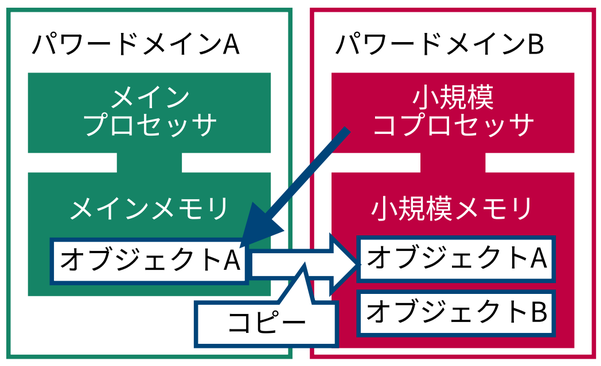
遠い場所にある領域をアクセス禁止にしておいて，
適切なタイミングで近い場所にコピーする

省電力性

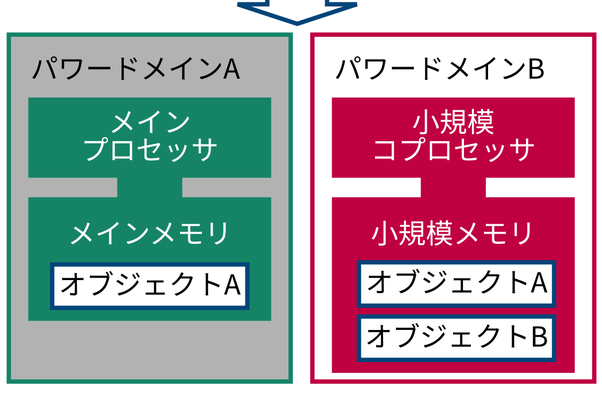
パワードメインBが動作する



オブジェクトAを参照しようとする

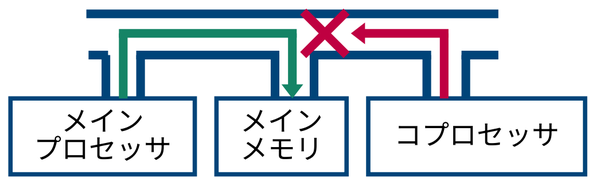


コピーのためにパワードメインAが
起動する



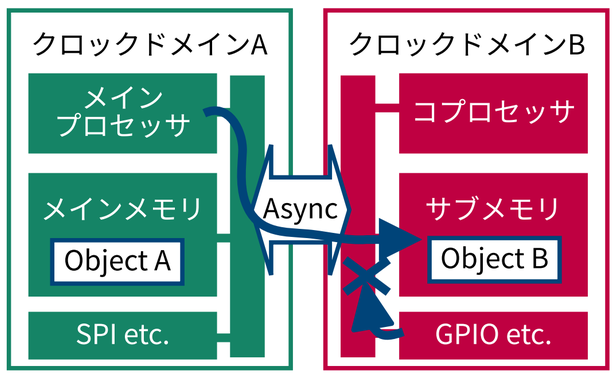
実時間性

バス接続のイメージ



特にメインメモリは同時アクセス不能
バスの同時アクセス数（レーン数）を
増やすと，消費電力は増加する

プロセッサごとにバスが分かれている
場合が多い

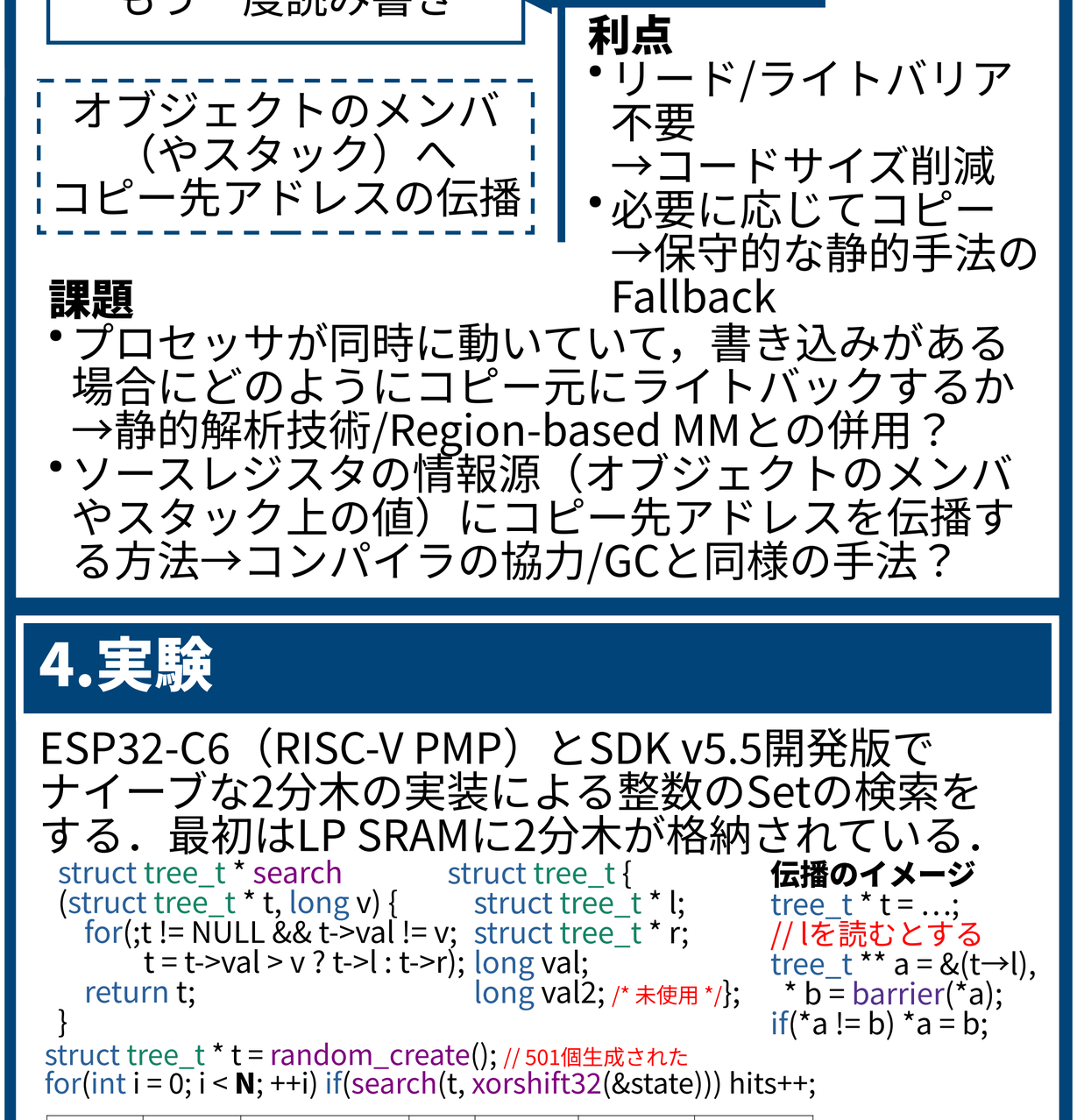


異なるバス上にある領域にアクセス
しようとすると，他のプロセッサの
処理に影響し，実時間性を損う

暇な時にアクセスすることが望ましい

3.例外によるアドレス変換手法

(1)コードサイズや(2)変換しない時のオーバヘッド
削減のために，例外を活用してバリアを実装する



利点
- リード/ライトバリア不要
  →コードサイズ削減
- 必要に応じてコピー
  →保守的な静的手法のFallback

課題
- プロセッサが同時に動いていて，書き込みがある場合にどのようにコピー元にライトバックするか
  →静的解析技術/Region-based MMとの併用？
- ソースレジスタの情報源（オブジェクトのメンバやスタック上の値）にコピー先アドレスを伝播する方法→コンパイラの協力/GCと同様の手法？

4.実験

ESP32-C6（RISC-V PMP）とSDK v5.5開発版で
ナイーブな2分木の実装による整数のSetの検索を
する．最初はLP SRAMに2分木が格納されている．

```
struct tree_t * search
(struct tree_t * t, long v) {
  for(;t != NULL && t->val != v;
        t = t->val > v ? t->l : t->r);
  return t;
}
```

```
struct tree_t {
  struct tree_t * l;
  struct tree_t * r;
  long val;
  long val2; /* 未使用 */};
```

伝播のイメージ

```
tree_t * t = …;
// lを読むとする
tree_t ** a = &(t→l),
  * b = barrier(*a);
if(*a != b) *a = b;
```

```
struct tree_t * t = random_create(); // 501個生成された
for(int i = 0; i < N; ++i) if(search(t, xorshift32(&state))) hits++;
```

メモリ	変換方法	アドレス変換表	伝播	コード[B]	実行時間比	実行時間比
メインメモリ上で2分木を全て作成				22	1	1
メインメモリにコピーしないLP SRAMのみ					10.06	10.15
LP SRAM	バリア	配列	なし	**40**	**7.24**	**3.95**
LP SRAM	バリア	配列	あり	64	7.98	4.66
LP SRAM	バリア*	配列	なし	64	7.52	4.07
LP SRAM	バリア*	配列	あり	90	6.98	3.12
LP SRAM	バリア	2分木連想配列	なし	40	18.00	11.64
LP SRAM	バリア	2分木連想配列	あり	64	18.88	4.13
LP SRAM	バリア*	2分木連想配列	なし	64	13.24	12.48
LP SRAM	バリア*	2分木連想配列	あり	90	12.66	3.19
LP SRAM	MPU	配列	なし	**22**	**24.32**	**20.31**
LP SRAM	MPU	配列	あり	38	10.92	3.08
LP SRAM	MPU	配列	HINT	36(ASM)	11.31	**3.01**
LP SRAM	MPU	2分木連想配列	なし	22	43.26	34.6
LP SRAM	MPU	2分木連想配列	あり	38	20.43	3.18
LP SRAM	MPU	2分木連想配列	HINT	36(ASM)	20.64	3.07

コードはsearch関数のサイズ

読み書き回数がある程度あれば速くなる

(1)search関数のコードサイズは変化なし．基本的に遅い

(2)伝播させることで読み書きが多ければ最速になる

バリア*は，アドレスの
範囲検査のみを
インライン展開した．

N=128 **N=2^{17}**

参照外し回数	1578	1.5×10^6
コピー回数	307	501

今後の予定と関連研究

- 省電力コプロセッサ向けmruby/cへの搭載
- 静的解析との組み合わせ – 静的言語処理系
  - 動的手法が役に立つ場面があるのか？

[A. Appel PLDI’88][R. Shaw CSL-TR-87-323] [B.Zorn CU-CS-494-90]
GCのバリアに活用. ZornはOSがある環境ではオーバヘッド大という追試.
[O. Avissar CASES’01] グローバル変数, スタックに対する最適化
[M. Meyer ISMM’06]ハードウェアによるGCのバリアをサポート
[J. Ceng DAC’08]ヘテロマイクロアーキに対する最適化, 並列化
[F. Lin ASPLOS’12]ソフトウェア分散メモリによる省電力コプロ利用 (C)