

# Virtual Secure Platform: A Five-Stage Pipeline Processor over TFHE

京都大学 佐藤研 M1 松岡 航太郎

五十嵐研 M1 伴野 良太郎

岡部研 M1 松本 直樹

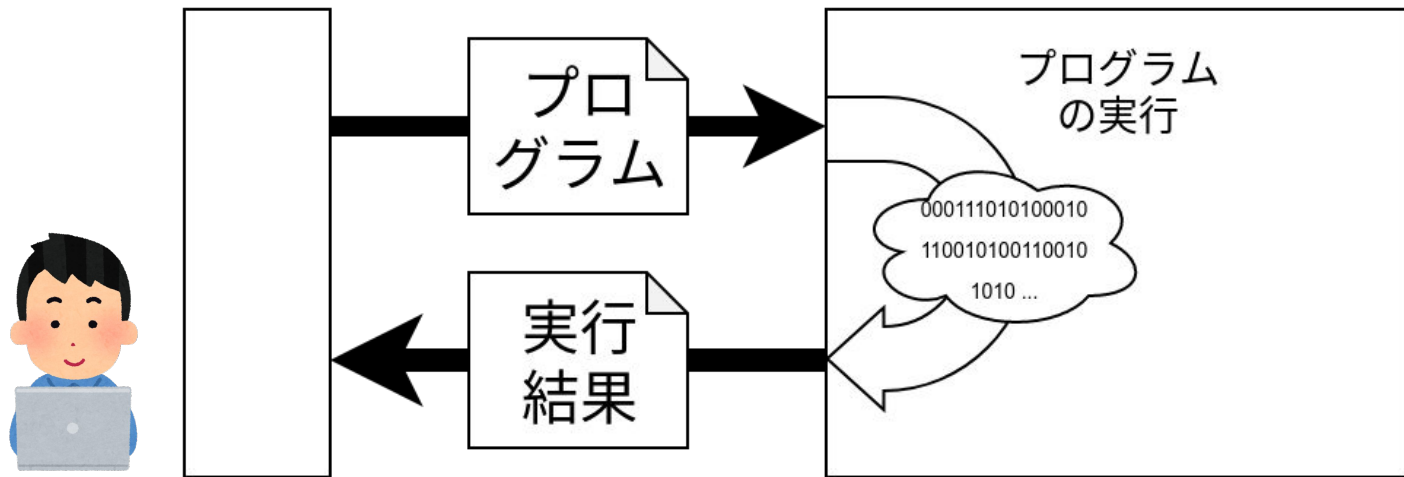
佐藤 高史 教授

Song Bian 助教

# 通常のクラウドコンピューティング

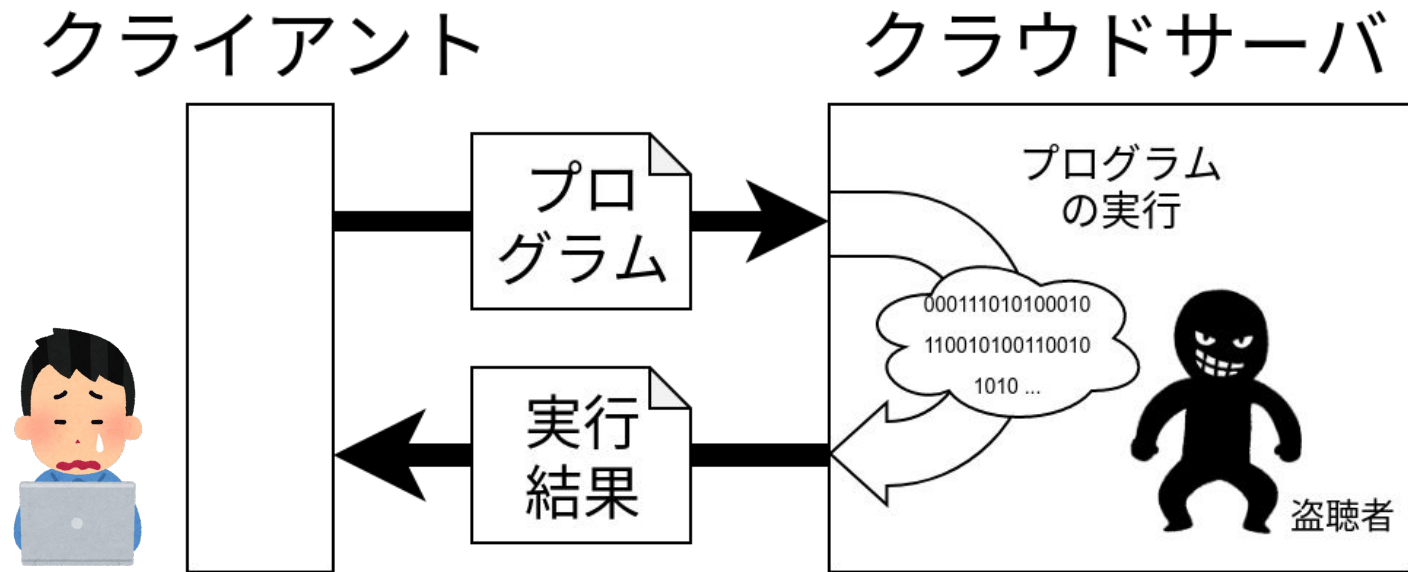
クライアント

クラウドサーバ



プログラムは**平文**で実行される

# 通常手法の問題点

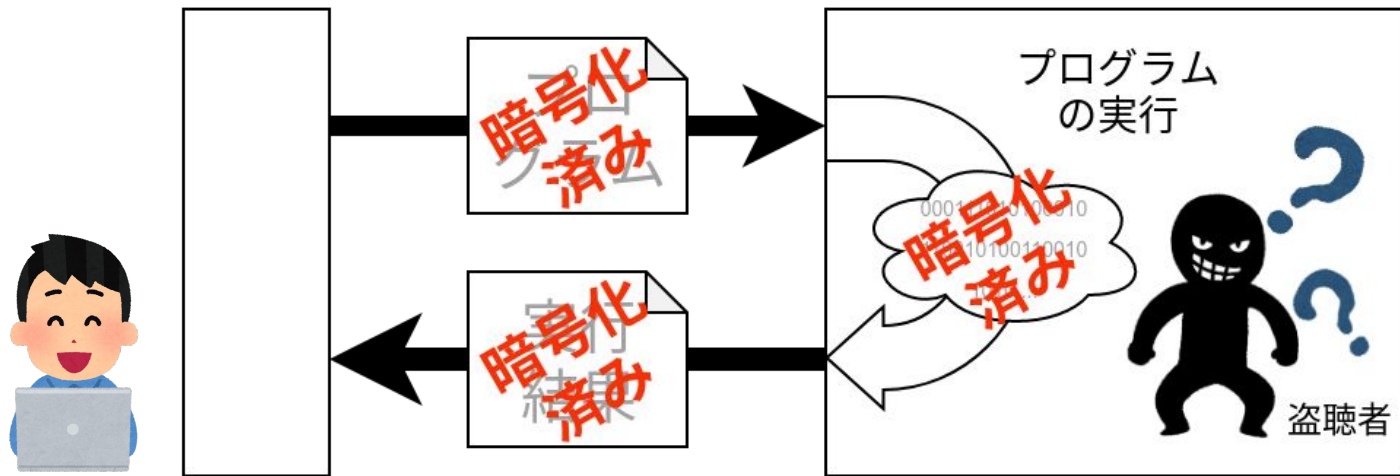


悪意ある第三者によって**盗聴される恐れ**  
(ハードウェアのバス信号を読むなど)

# 提案手法: Virtual Secure Platform(VSP)

クライアント

クラウドサーバ



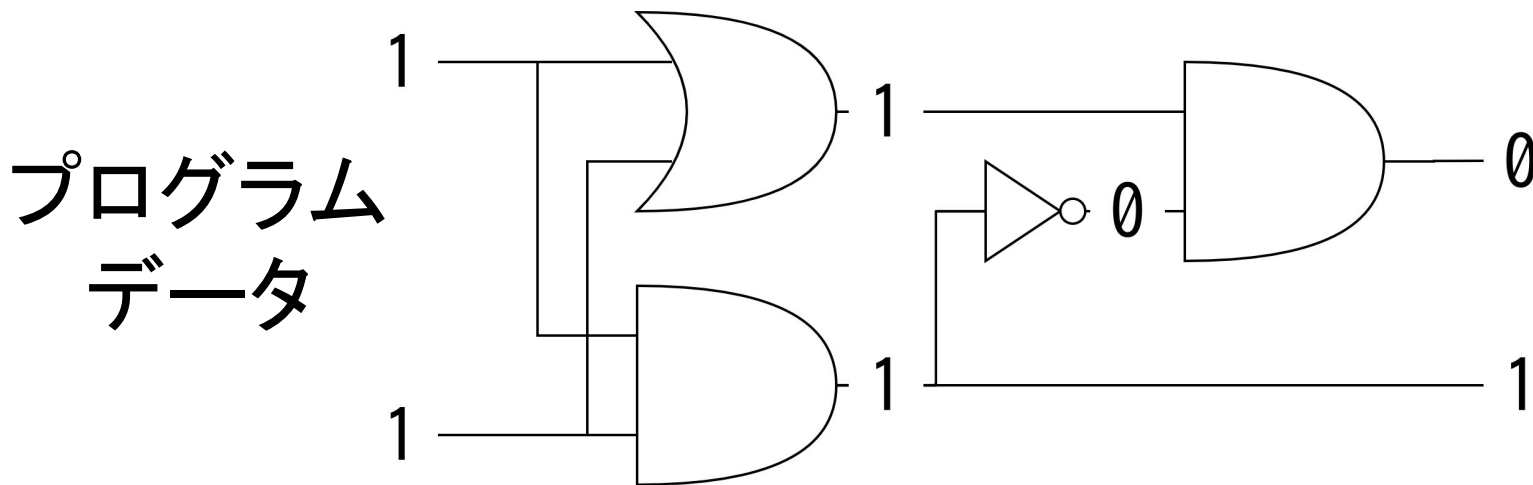
プログラムを含めすべて暗号化した状態で実行



**盗聴自体を無効化する**

# 暗号化されたプログラムを実行するには？

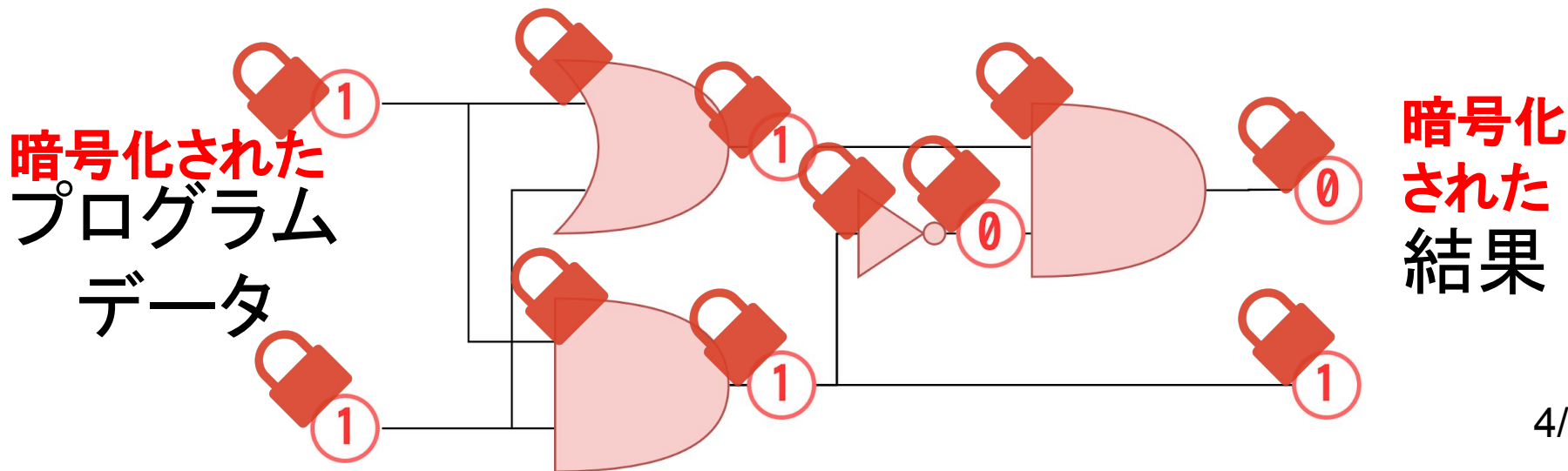
- プロセッサ = 論理回路 = 論理ゲートのグラフ
- 上の問いは“論理ゲートをどうやって暗号化されたbitに対して評価するか？”という問いに落とせる
- それを達成する方法の一つが完全準同型暗号
  - 我々は TFHE という論理回路に特に適したものを使っている



結果

# 暗号化されたプログラムを実行するには？

- プロセッサ = 論理回路 = 論理ゲートのグラフ
- 上の問いは“論理ゲートをどうやって暗号化されたbitに対して評価するか？”という問いに落とせる
- それを達成する方法の一つが完全準同型暗号(順序回路に適している)
  - VSPは TFHEという論理回路に特に適したものを使っている



# Our Contribution

1. 高級言語との互換性
2. 高速化
3. オープンソース実装

# Our Contribution

## 1. 高級言語との互換性

- セキュリティと利便性の間にはトレードオフがある
- **RISC-VベースのISA** と LLVM9ベースの**Cコンパイラ**

## 2. 高速化

## 3. オープンソース実装



# Our Contribution

## 1. 高級言語との互換性

- セキュリティと利便性の間にはトレードオフがある
- **RISC-VベースのISA** と LLVM9ベースの**Cコンパイラ**

## 2. 高速化

- 完全準同型暗号の演算は遅いので特別な考慮が必要
- lyokan (FHE論理回路評価エンジン)による並列性の利用
- **CMUX Memory**(TFHEでの最適化されたメモリの実現法)の提案と実装
- これらを合わせることでVSP はFURISCに比して**1600倍**高速化

## 3. オープンソース実装

# Our Contribution

## 1. 高級言語との互換性

- セキュリティと利便性の間にはトレードオフがある
- **RISC-VベースのISA** と LLVM9ベースの**Cコンパイラ**

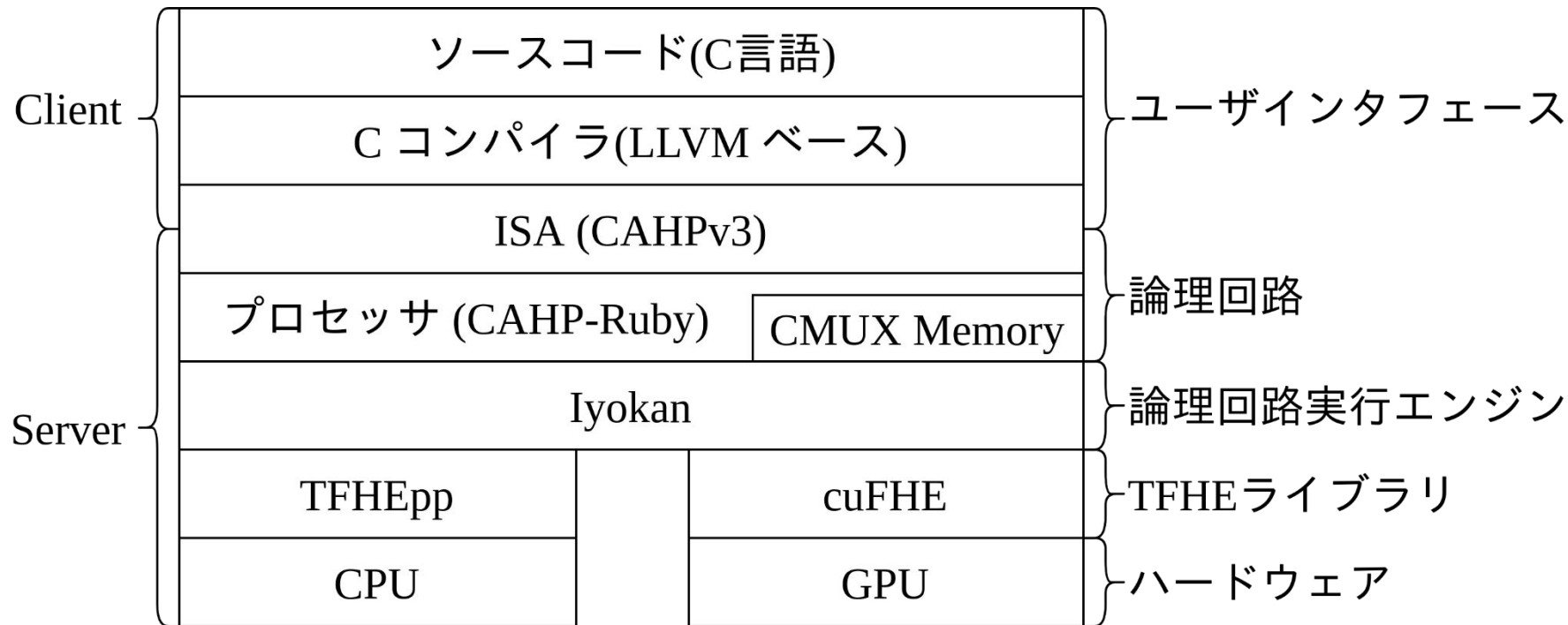
## 2. 高速化

- 完全準同型暗号の演算は遅いので特別な考慮が必要
- lyokan (FHE論理回路評価エンジン)による並列性の利用
- **CMUX Memory**(TFHEでの最適化されたメモリの実現法)の提案と実装
- これらを合わせることでVSP はFURISCに比して**1600倍**高速化

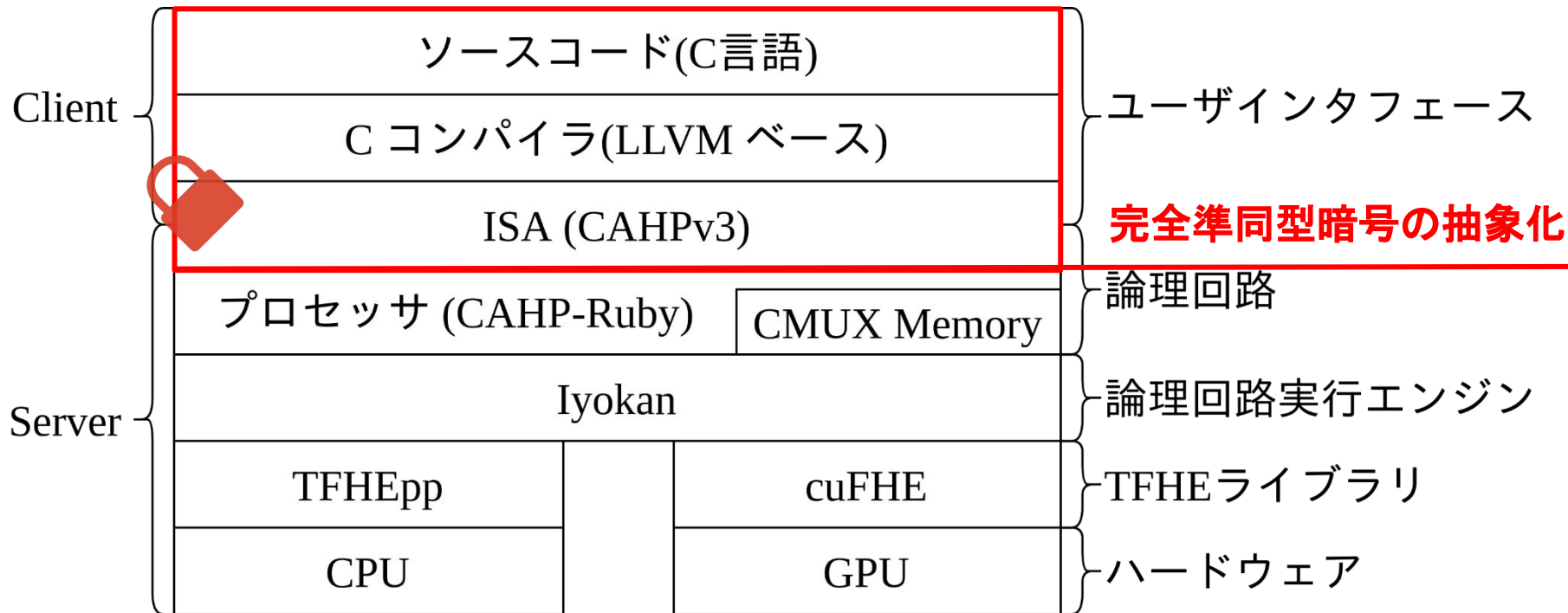
## 3. オープンソース実装

- 結果の再現性と実装の検証性を提供
- Available on **GitHub** <https://github.com/virtualsecureplatform/kvsp>

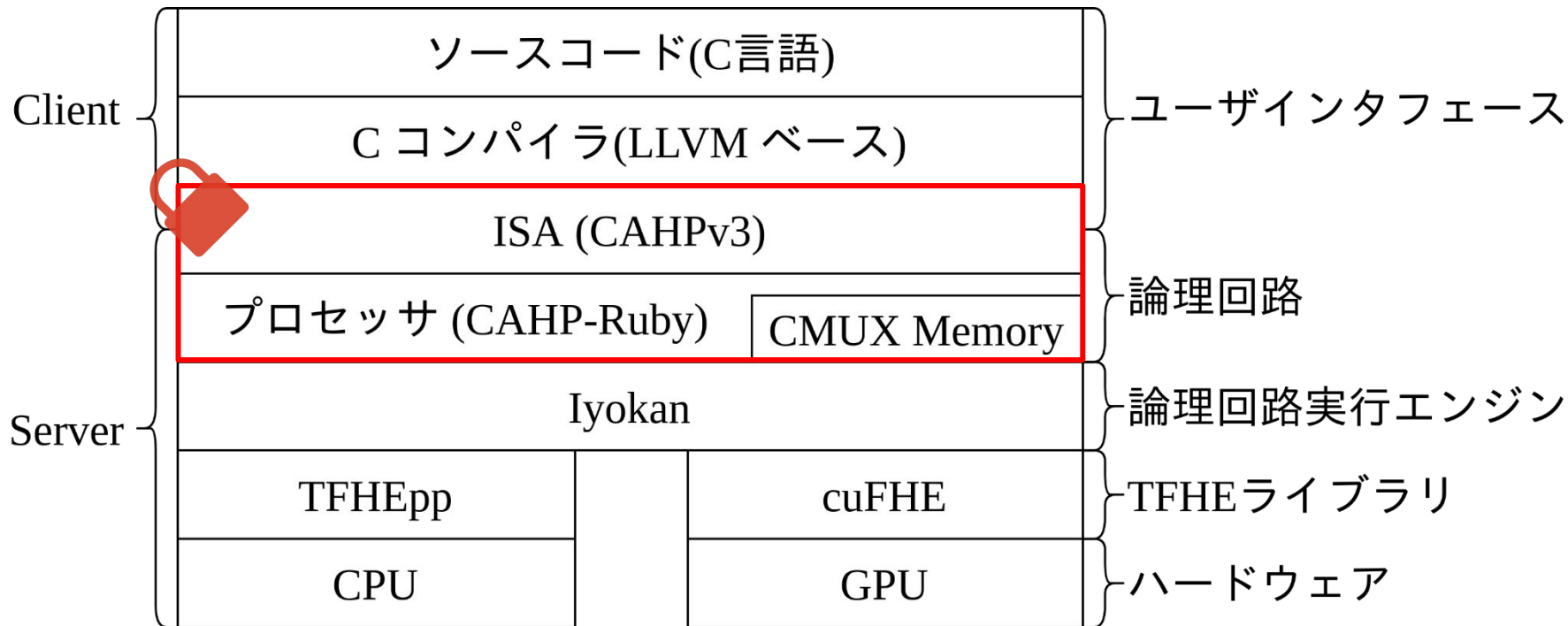
# アーキテクチャ概略図



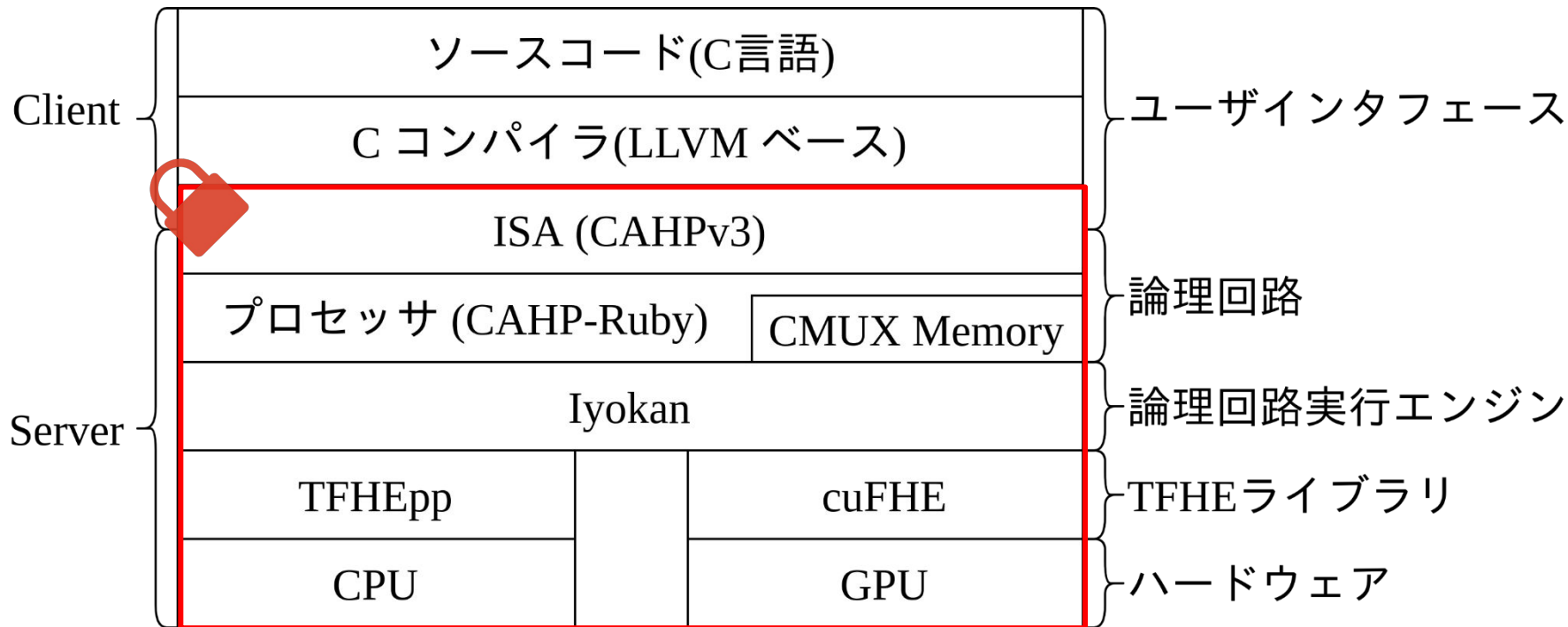
# アーキテクチャ概略図



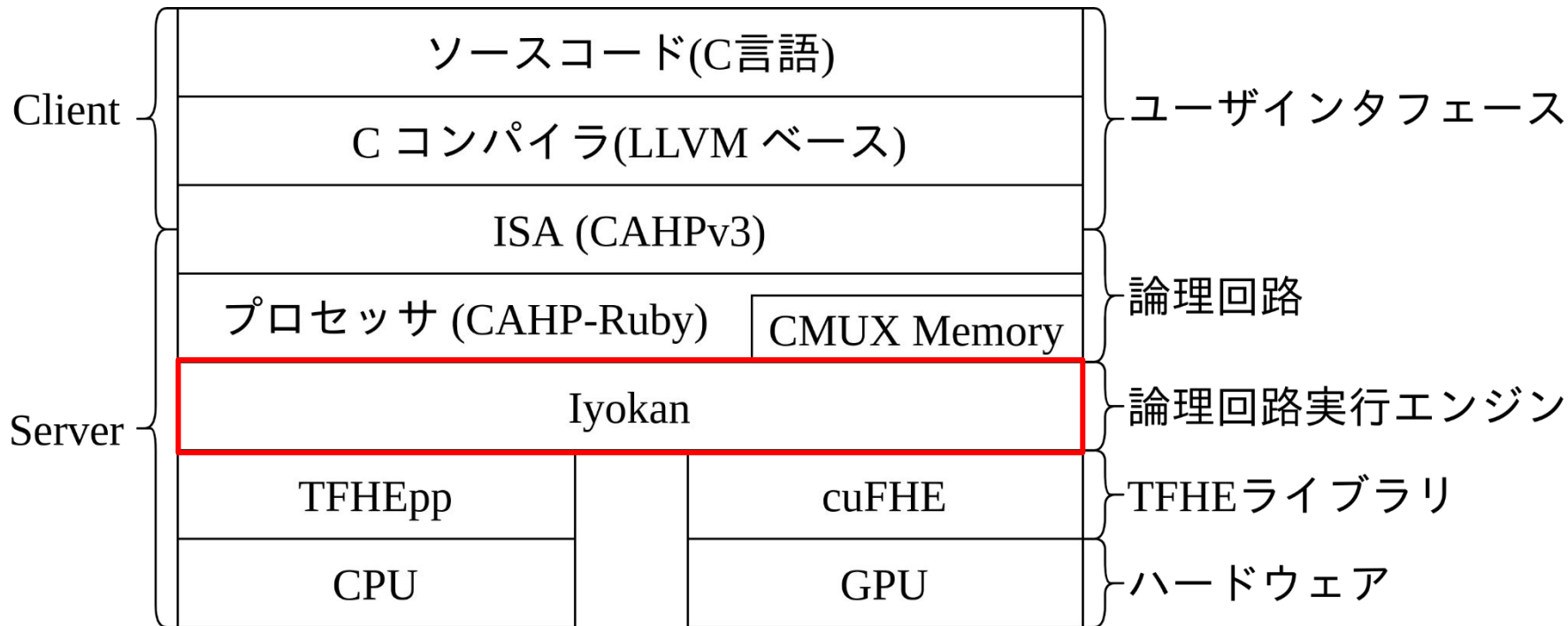
# アーキテクチャ概略図



# アーキテクチャ概略図



# アーキテクチャ概略図

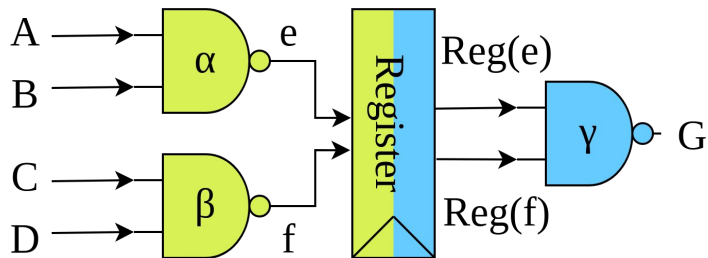
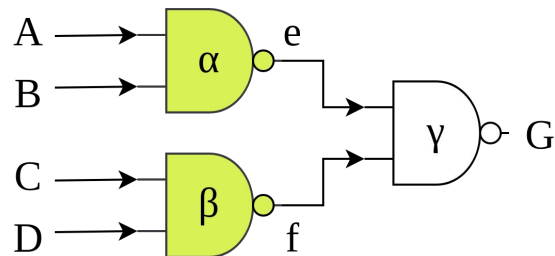


# ゲートレベルの並列性

- ゲートは並列に実行可能
  - 依存があると実行不能

## パイプラインによる並列性

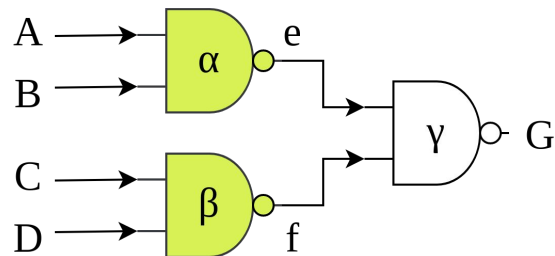
- 回路を同時に評価可能な複数の部分に分割





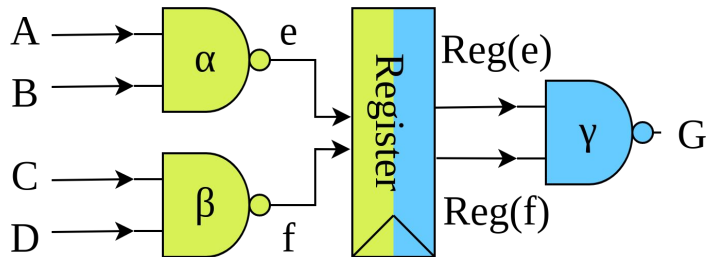
# ゲートレベルの並列性

- ゲートは並列に実行可能
  - 依存があると実行不能



# パイプラインによる並列性

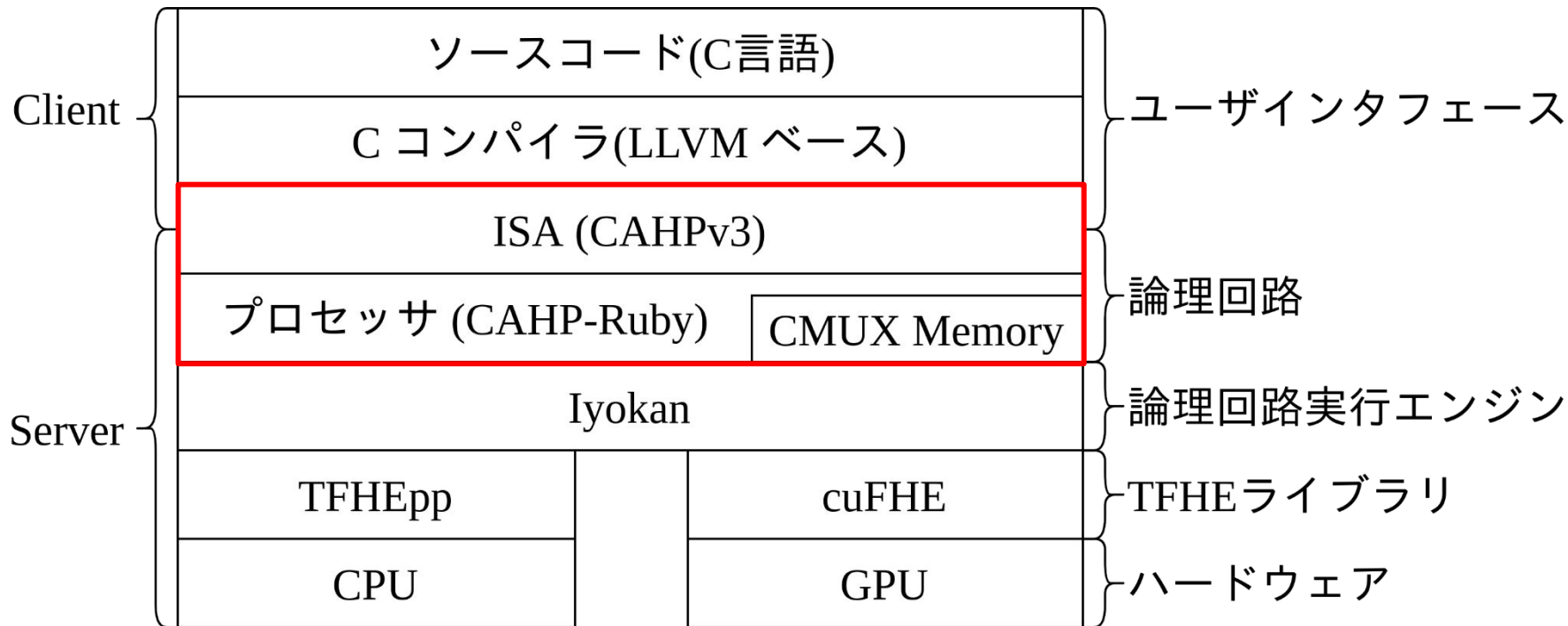
- 回路を同時に評価可能な複数の部分に分割



# lyokanにおける並列性の利用

- 物理的な実行環境に依存している
  - 本質的に並列に動作する物理的な論理回路とは違う
  - 強力な並列計算能力とスケジューリングが必要(lyokanの役割)
  - lyokanはマルチコアCPUとマルチGPUの両方に対応している

# アーキテクチャ概略図



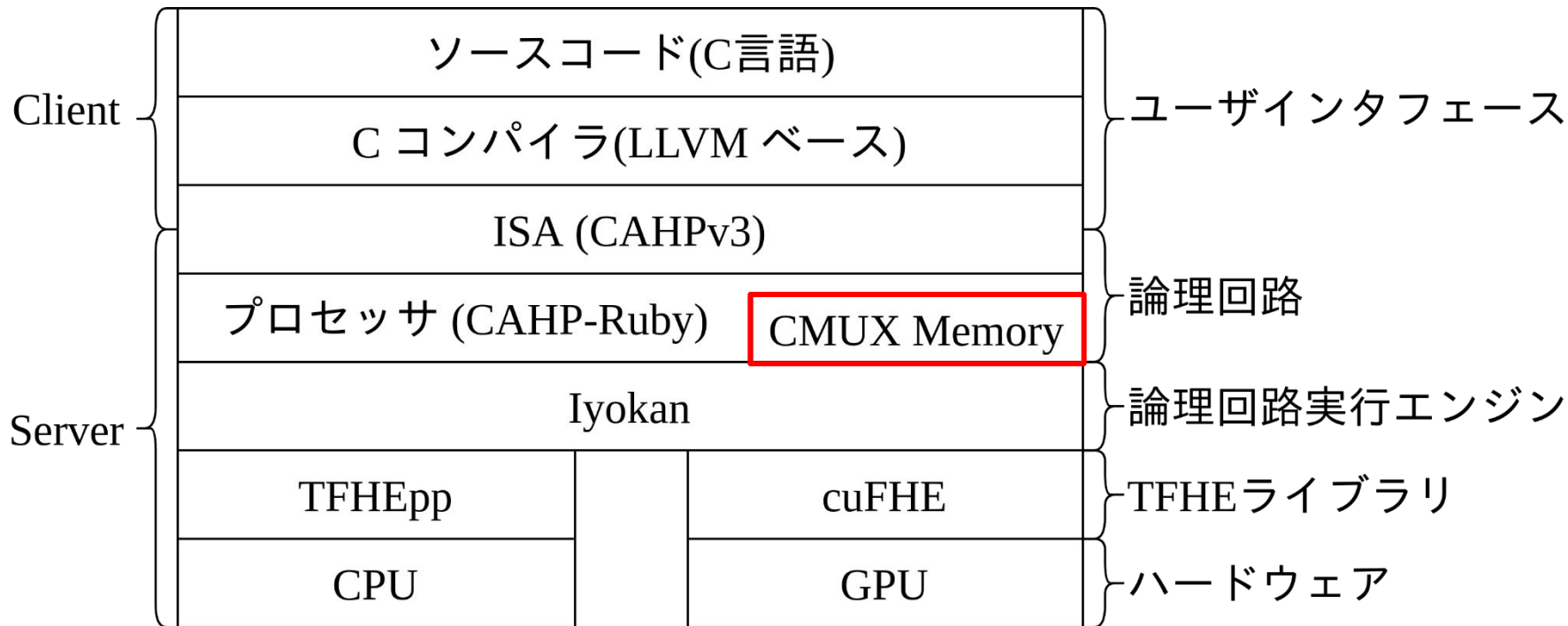
# 準同型暗号上の論理回路設計で考慮すべきこと

- もし演算器が1つしか存在しなかったら
  - 論理ゲートが少ない方がより高速
- もし演算器が無限に存在したら
  - 物理的な論理回路と同様に設計すればよい
- 1台のマシンで用意できる並列性には限りがある
  - ヒューリスティックとしては論理ゲートの総数は少ないほうが良い
  - 同じゲート数なら並列性が高いほうが良い

# 準同型暗号上の論理回路設計で考慮すべきこと

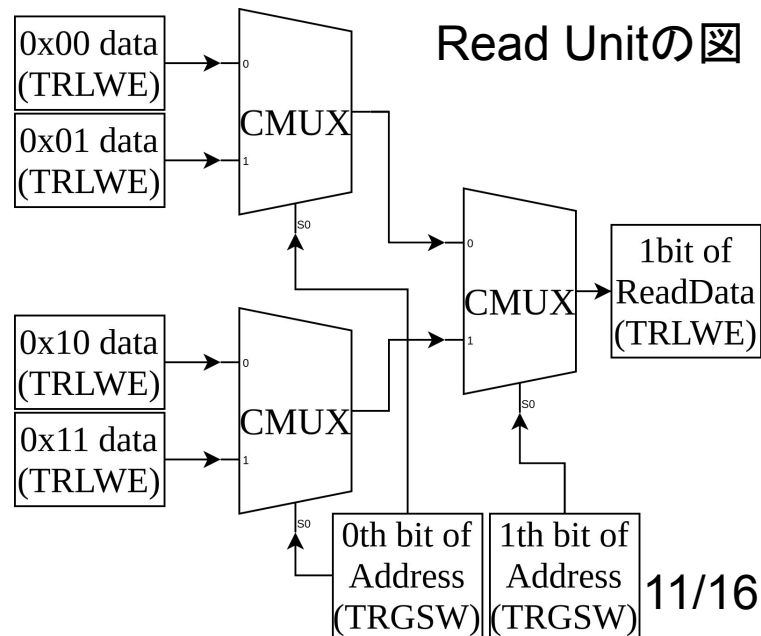
- もし演算器が1つしか存在しなかったら
  - 論理ゲートが少ない方がより高速
- もし演算器が無限に存在したら
  - 物理的な論理回路と同様に設計すればよい
- 1台のマシンで用意できる並列性には限りがある
  - ヒューリスティックとしては論理ゲートの総数は少ないほうが良い
  - 同じゲート数なら並列性が高いほうが良い
- ISA とプロセッサの設計では主にゲート数を減らすように設計
  - 16bitデータ長、24/16bit可変長命令でバス幅とメモリの回路を小さく
  - パイプライン化で並列性を向上
  - パイプラインありで約2,400ゲート、なしで約2,000ゲート

# アーキテクチャ概略図



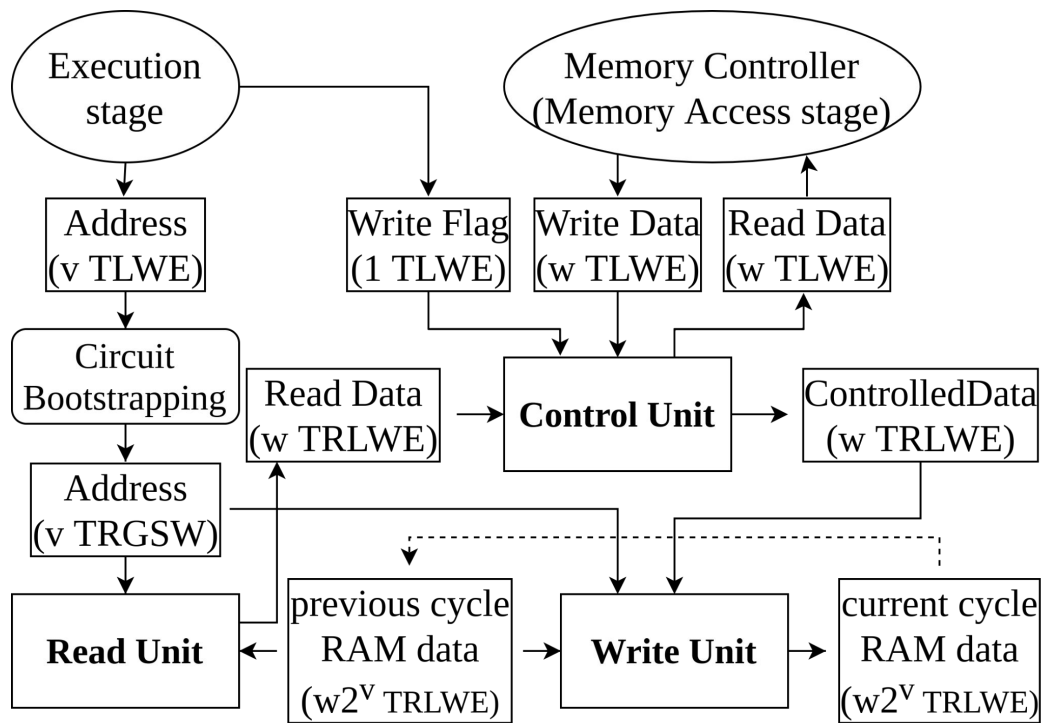
# CMUX Memory

- TFHEのLHE(Leveled Homomorphic Encryption)モードを利用したメモリ
  - 回路の深さに上限があるがメモリ回路の深さはアドレス幅(9bit)分程度
- CMUXはLHEモードでMUXを評価する方法
  - 完全準同型暗号としてのMUXより  
**約1000倍高速**
  - セレクタに入る暗号文の形式がデータと異なる
    - 出力がセレクタに入れない



# CMUX RAM

- 暗号文の種類を変換する処理が重い
  - Circuit Bootstrapping
  - 論理ゲートの10倍ほど
  - ROMの計算量はアドレス幅に比例
  - RAMはサイズに比例



v: アドレス幅 w: データ幅

# 速度評価(パイプライン)

- 実験環境
  - CMUX Memoryは使っている
  - テストプログラム: ハミング距離の計算
  - ケース 1: AWS p3.16xlarge (64 vCPUs, 488GB RAM, 8 V100 GPUs)
  - ケース 2: さくら高火力 (16vCPUs, 128GB RAM, 1 V100 GPU)
- ケース2では並列実行可能なゲート数に対し並列計算能力が不足

ケース	1	2
w/ Pipeline (CAHP-ruby)	<b>0.8</b> s/cycle	1.7 s/cycle
w/o Pipeline (CAHP-pearl)	1.5 s/cycle	2.4 s/cycle



# 速度評価(CMUX Memory)

- 並列計算能力が高いときはCMUX Memoryによる影響が小さい
  - メモリはある程度並列性がある
- 並列計算能力が低いときはサイクル当たりの時間に直結

ケース	1	2
w/ CMUX Memory	<b>0.8</b> s/cycle	1.7 s/cycle
w/o CMUX Memory	1.3 s/cycle	6.4 s/cycle

# FURISCとの比較

- FURISC
  - State-of-the-Art のFHE (Smart-Vercauteren) 上で構成されたプロセッサ
  - Subtract Branch if Negative (SBN)の1命令だけをサポート
  - コンパイラがない (VSPはC コンパイラがある)
- VSPの方が約**1600**倍高速
  - lyokanによる並列性の利用とCMUX Memory
- VSPの実装はオープンソース

Name	sec./cycle	実装
VSP	<b>0.8</b>	公開
FURISC	1278 (est.)	非公開

# 結論

- VSPはプログラムを秘匿したままでの計算処理の委託を可能にする
- FHEによる論理回路の評価では並列性を利用することが重要
  - ゲートレベルとアーキテクチャレベルの両方の並列性を考慮に入れる
    - 並列計算能力を使い切る程度の並列性
  - 強力な並列計算能力とそれを制御するスケジューラが必要
- CMUX Memoryはメモリの高速化に寄与する

ご清聴ありがとうございました