量子AIシミュレーターと量子SDKの開発

鈴木鉄兵, 博士（工学）
SCSK株式会社

2023年8月3日 @第3回量子ソフトウェアワークショップ
（東京大学理学系研究科量子ソフトウェア寄付講座）
15:25–16:05

登壇者紹介

鈴木 鉄兵
博士（工学）

- 早稲田大学 理工学部 電気電子情報工学科卒業
- 同大学大学院 理工学研究科 生命理工学専攻 博士後期課程修了、博士（工学）
- 博士論文のテーマ：第一原理分子動力学シミュレーション

- （学校法人）早稲田大学・助手、客員講師（常勤）
- （国研）物質・材料研究機構・ポスドク研究員
- （国研）理化学研究所・協力研究員
- （民間企業）複数のIT企業で受託業務等に従事、量子ベンチャー企業を経て、現職

2021年にはSCSKで量子チーム・リーダーとなり、プロジェクトのマネジメントを行いながら、市場動向調査、アルゴリズム研究、量子SDK・APIの開発、アプリケーション開発、論文発表やプレスリリースなどの業務に従事

専門分野

- 量子化学分野： 量子化学、電子状態計算、第一原理分子動力学シミュレーション
- 機械学習分野： マテリアルズ・インフォマティクス、画像処理技術、非線形回帰
- HPC分野: スーパーコンピュータ、ヘテロジニアス・コンピューティング、GPU、FPGA
- 量子コンピューティング分野： 量子AI、量子アルゴリズム、量子ソフトウェア、量子技術の社会実装

量子AIに関する論文：

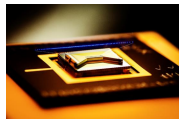
[1] T. Suzuki, T. Hasebe & T. Miyazaki. Quantum support vector machines for classification and regression on a trapped-ion quantum computer (2023) arXiv: 2307.02091
[2] T. Suzuki, T. Miyazaki, T. Inaritai & T. Otsuka. Quantum AI simulator using a hybrid CPU–FPGA approach. *Sci. Rep.* 13 (2023) 7735
[3] T. Suzuki & M. Katouda. Predicting toxicity by quantum machine learning. *J. Phys. Commun.* 4 (2020) 125012

本日の発表内容

1. 研究背景と概要

2. 量子AIシミュレーター

3. FPGA実装



4. 量子SDKと実機検証

5. まとめ

本講演では、IT企業（SIer）における
企業研究の立場から、量子コンピューティング
の社会実装に向けた取り組み内容を紹介する。

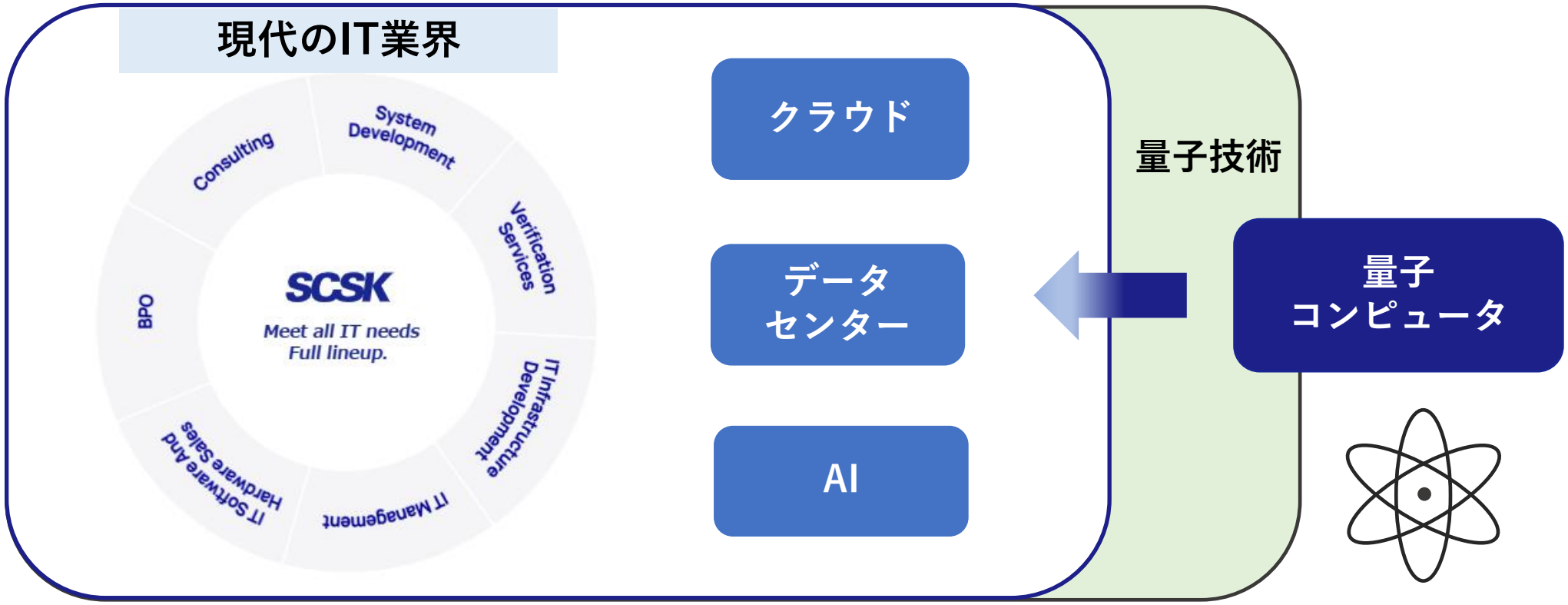
具体的には、量子カーネル法を中心に、FPGA
を活用した量子AIシミュレーター、
また、量子SDKの活用事例として、
Amazon Braketが提供するイオントラップ型
量子コンピューターIonQ Harmonyを
使って実機検証した結果についても紹介する。

登壇者：
鈴木 鉄兵（PM、量子アルゴリズム）

技術者（SCSK）：
大塚 高廣（FPGAアーキテクチャ設計）
稲荷平 駿稀（FPGAエンジニアリング、AI）
宮﨑 翼（SDK開発、量子プログラミング、AI）
長谷部 嵩（AIモデルの調整）

現代のIT業界における量子コンピューティングの可能性

クラウドコンピューティング、データセンター、AIは、現代のIT業界の最前線に位置している



量子コンピューティングは、高速または省エネなデータ処理を提供することで、
これらの技術を補完する可能性がある

量子コンピューティングの将来的なユースケース†

潜在市場

シミュレーション	最適化	機械学習	暗号・通信
製薬：創薬	金融：ポートフォリオ最適化	自動車：自動運転、AI	政府：暗号化と復号化
航空宇宙：計算流体力学	保険：リスク管理	金融：不正検知	民間：暗号化と復号化
化学：触媒設計	物流：ネットワーク最適化	IT：検索と広告最適化	
エネルギー：エネルギー変換	航空宇宙：ルート最適化		
金融：金融デリバティブの評価			

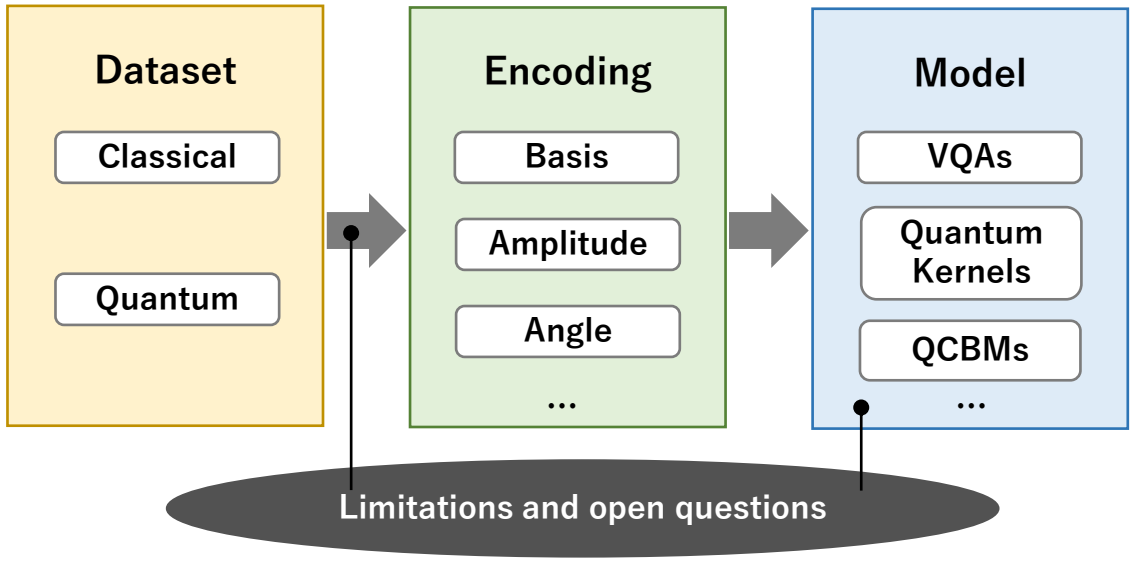
機械学習は様々な分野にインパクトをもたらす

†ボストンコンサルティンググループの資料をもとに作成
（Bobier, J.-F.; Langione, M.; Tao, E.; Gourevitch, A. *What happens when 'if' turns to 'when' in quantum computing?* (2021)
https://www.bcg.com/publications/2021/building-quantum-advantage

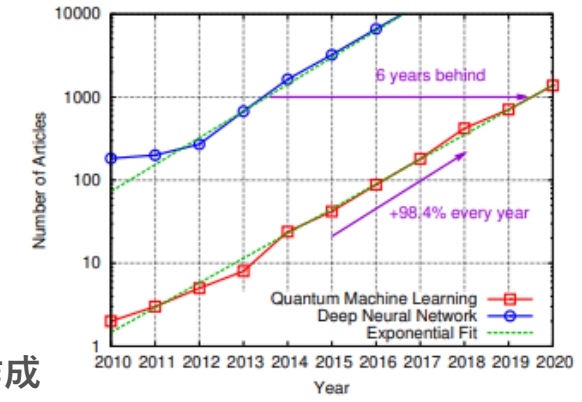
量子AIの現状

- 量子AIの場合、データセットとして量子データと古典データがあり得る
- データのエンコーディングには、様々な手法が提案されている：基底、振幅、角度など
- 代表的な量子AIアルゴリズム：変分量子アルゴリズムや量子カーネル法がある
- データのエンコーディングに関する制限、量子AIの拡張性・優位性など、未解決な点が多い
- 量子AIに関する論文の数は、2010年から確実に増加している
- 応用領域：高エネルギー物理、ベンチマーク、画像、化学、サイバーセキュリティなど

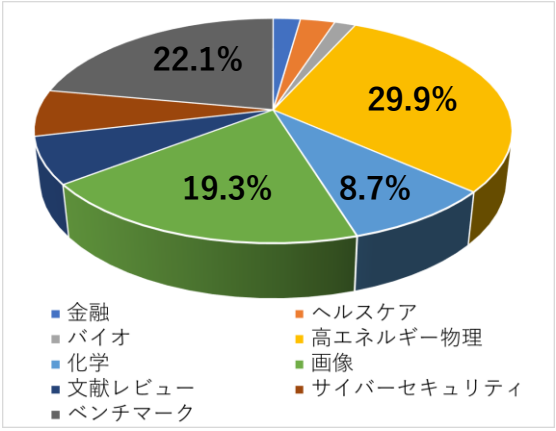
(a) 量子AIの概要



(b) 量子AIの論文数



(c) 応用領域の内訳



出所：
a), c): Gujju, Y. *et al.* arXiv: 2307.00908 (2023)を参考に著者が作成
b): Koike-Akino, T. *et al.* arXiv: 2205.08590 (2022)

なぜ量子カーネル法に注目したのか？

カーネル法とは、線形分離できないような非線形のデータを高次元空間に写像し、その空間で線形分離可能になるように学習する方法

重要なアルゴリズムの一つ

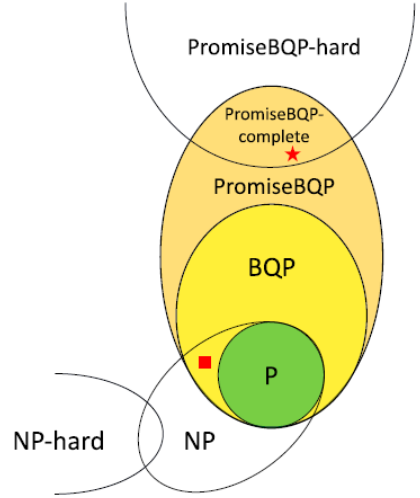
NISQアルゴリズム（今回の話題）だけでなく、誤り訂正のある量子コンピュータにおいても、量子カーネル法の概念は有用あると期待される

理論が整備されている

数学的な理論が整備されており、カーネル法の性質や学習アルゴリズムの収束性などが厳密に評価されている

機械学習ライブラリの活用

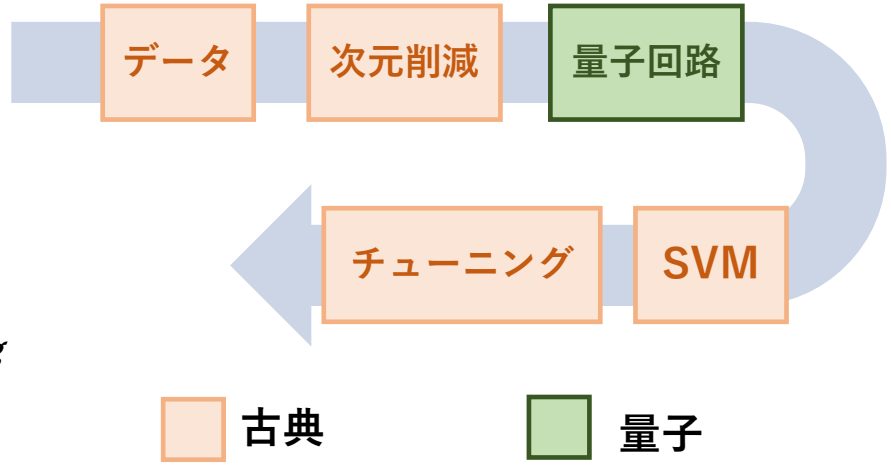
今回は量子カーネルのみが量子計算であり、その他のタスクは既存の機械学習ライブラリを利用できる（今いるAI人材の活用）



Jäger, J. & Krems, R. V. *Nat. Commun.* 14 (2023) 576.

- ヒルベルト空間
- 内積、ノルム
- 特徴写像
- 再生核空間の理論
- etc

書籍：Schölkopf, B. & Smola, A. J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* (MIT Press , Cambridge, MA, 2002)

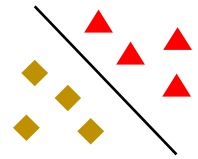


量子サポートベクターマシンのワークフロー

サポートベクターマシン： 機械学習の一つで、高次元の特徴空間に写像し、データを識別する方法

学習フロー

$$K(\boldsymbol{x}, \boldsymbol{x}') = |\langle \phi(\boldsymbol{x}) | \phi(\boldsymbol{x}') \rangle|^2$$



教師データ → 主成分分析（次元削減） → カーネル計算 → マージン最適化 → パラメータ、サポートベクター

特徴関数を量子回路で顕に表現し、
量子カーネル行列を生成する

推論フロー

$$K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}\right)$$

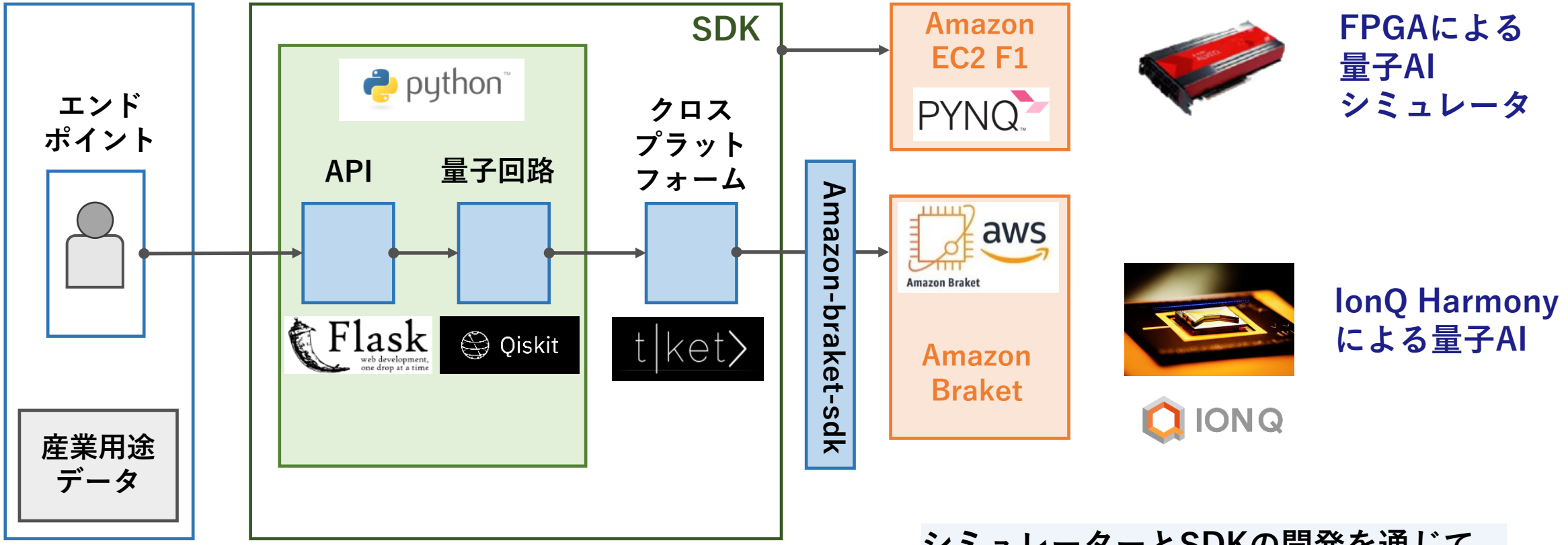
テストデータ → 主成分分析（次元削減） → カーネル計算 → 分類

$$y = \mathrm{sgn}\left(\sum_i y_i \alpha_i^* K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}\right) + b^*\right)$$

テストデータとサポートベクター
を用いて計算する

AWSクラウドサービスを活用する

Amazon Elastic Compute Cloud（EC2）F1インスタンスやAmazon Braket（QPU）などのAWSクラウドサービスを使用して、量子AIを探索する



シミュレーターとSDKの開発を通じて、将来の量子時代に備える

2. 量子AIシミュレーター

SCSK
夢ある未来を、共に創る。

1. 研究背景と概要

2. 量子AIシミュレーター

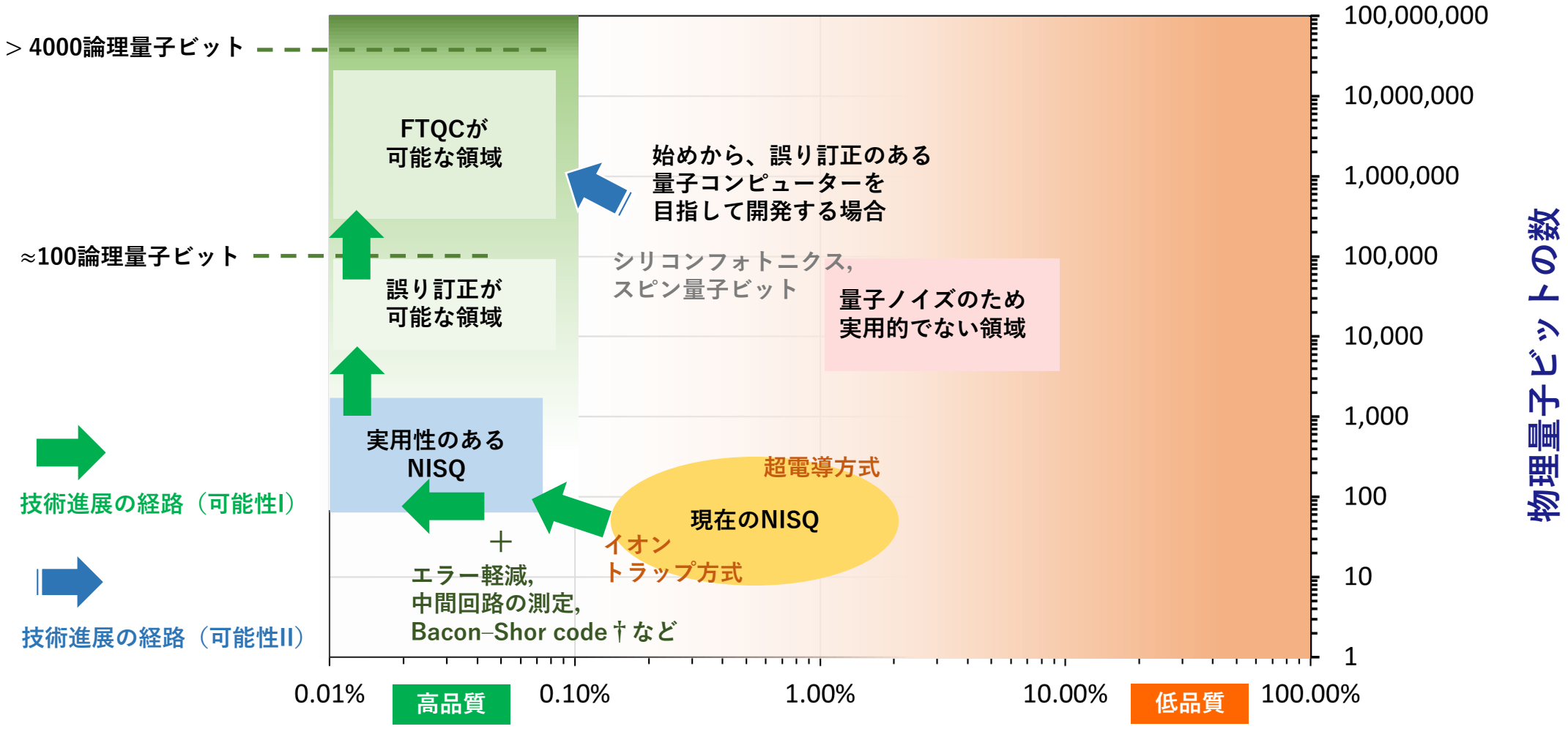
3. FPGA実装

4. 量子SDKと実機検証

5. まとめ

2. 量子ハードウェアの現状と今後：実用化へ至る経路

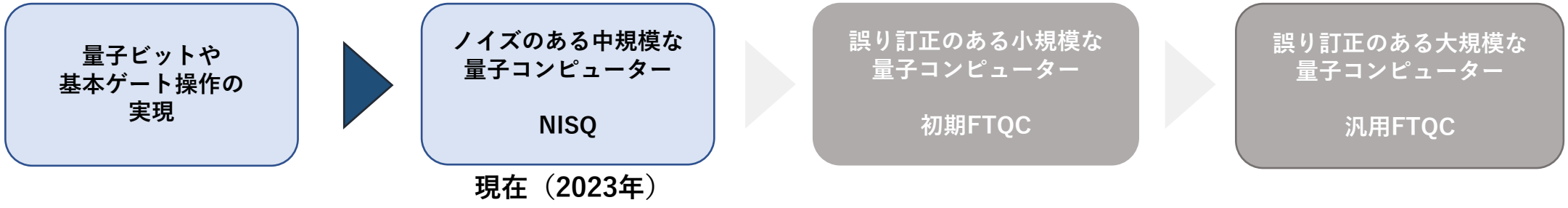
NISQ：ノイズのある中規模な量子コンピューター；　FTQC：誤り訂正のある量子コンピューター



出所：Ezratty, O. arXiv:2305.09518 (2023)を参考に筆者が見解を追加して作成

† Egan et al. *Nature* 598 (2021) 281–286

NISQの現状とシミュレーターの必要性



NISQ（Noisy Intermediate-Scale Quantum）ハードウェアの課題

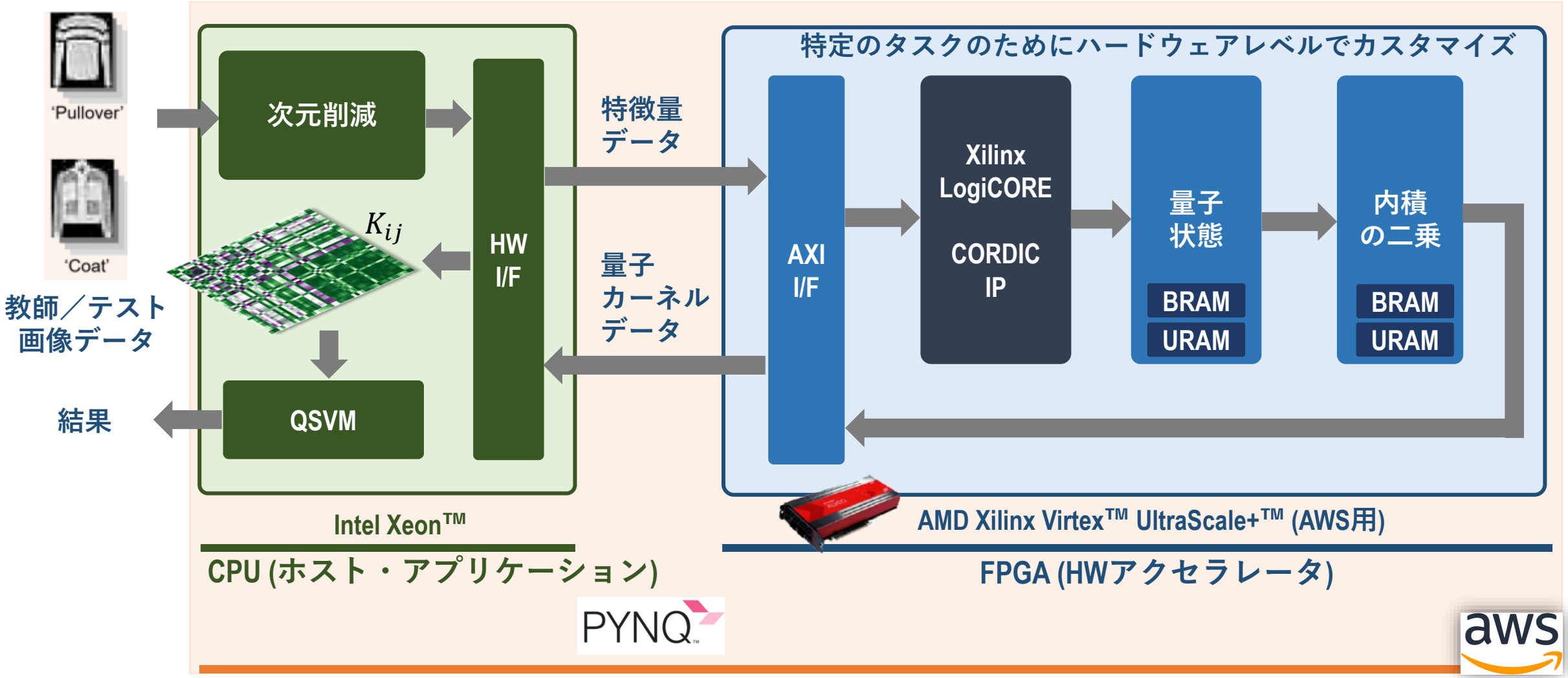
量子ビットのエラー	量子状態は繊細であり、量子ビットやゲート操作のエラー率が高く、深い量子回路が実行できない
スケーラビリティ	物理量子ビット数は数十から数百であり、ノイズの影響により「有効な」量子ビットはさらに少ない
アクセスと費用	一部の研究機関や企業しかアクセスできないハードウェアも多く、クラウド利用のコストも高い

量子コンピュータ・シミュレーターが必要な理由

ノイズレス	ノイズの無い状態でシミュレーションできるため、量子アルゴリズムを評価しやすい
低コスト	古典コンピューターであれば、PCやクラウド利用のコストは低く、試行錯誤やテストができる

汎用シミュレータを作るのではなく、ユースケースや社会実装を起点としてシミュレーターを開発する

Amazon EC2 F1インスタンスを活用した量子回路シミュレーター



Suzuki *et al*. *Sci. Rep*. **13**, 7735 (2023)

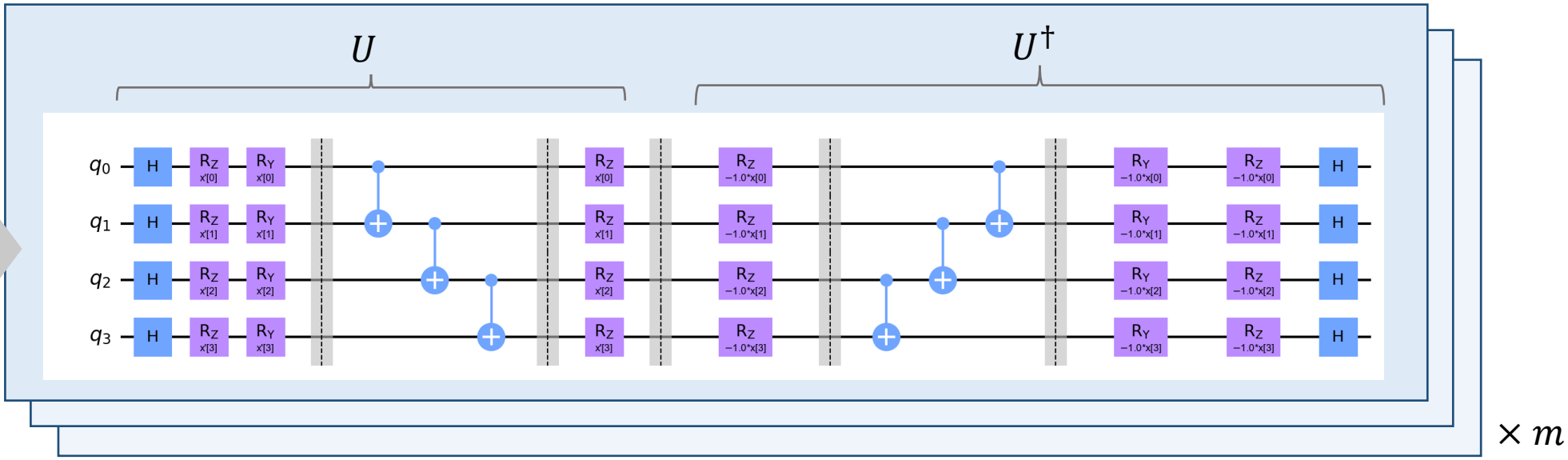
ブロック分割と浅い量子回路による量子カーネルの計算

（次元削減後の）ベクトルxをmのブロックに分割し、それぞれの特徴写像で内積を計算する

$$\boldsymbol{x} = \{ \boldsymbol{s}_1 , \boldsymbol{s}_2 , \boldsymbol{s}_3 , \cdots , \boldsymbol{s}_m \}$$

$\boldsymbol{s}_b$

$b = 1, 2, \cdots, m$



$\times m$

- この量子回路部分を古典コンピューター（FPGA）でシミュレーションする
- 特徴写像：各ブロック内では量子もつれを作るが、ブロック間では量子もつれを作らない

$$K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right) = \prod_{b=1}^{m} \left| \left\langle \psi_b\left(\boldsymbol{s}_b^{(i)}\right) \middle| \psi_b\left(\boldsymbol{s}_b^{(j)}\right) \right\rangle \right|^2$$

CPU-FPGAハイブリッド・アプローチによる量子SVM

画像データをPCAにより次元削減を行う

▼

データはCPUからFPGAにPCIeで転送される

▼

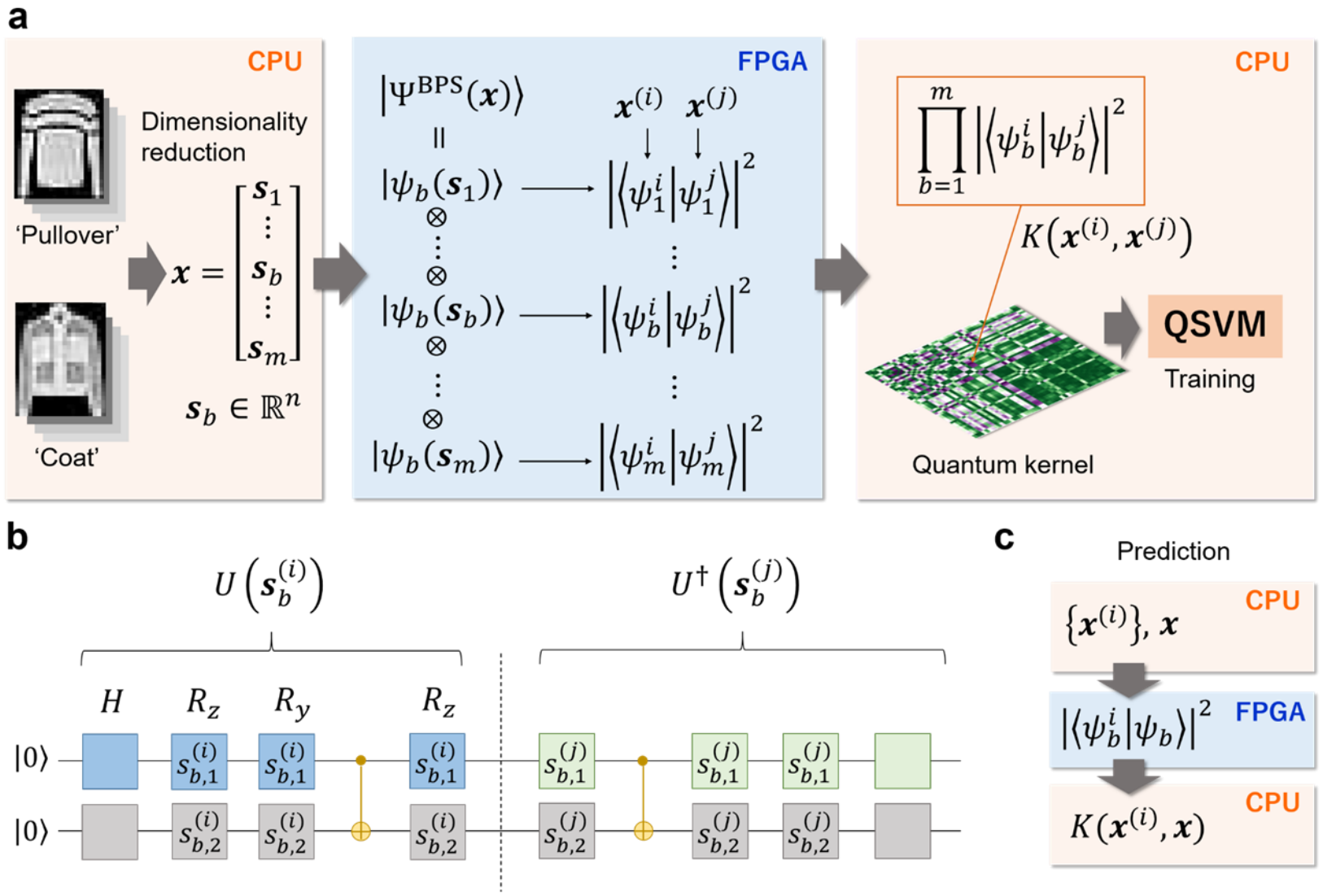
浅い固定の量子回路を設計した（この部分をFPGA実装でアクセラレーション。量子もつれのための量子ビット数は6まで対応）

▼

FPGAで計算した内積をCPUに転送し、量子カーネル行列を構築する

▼

得られた量子カーネル行列を使って、量子サポートベクターマシン（QSVM）を実行する

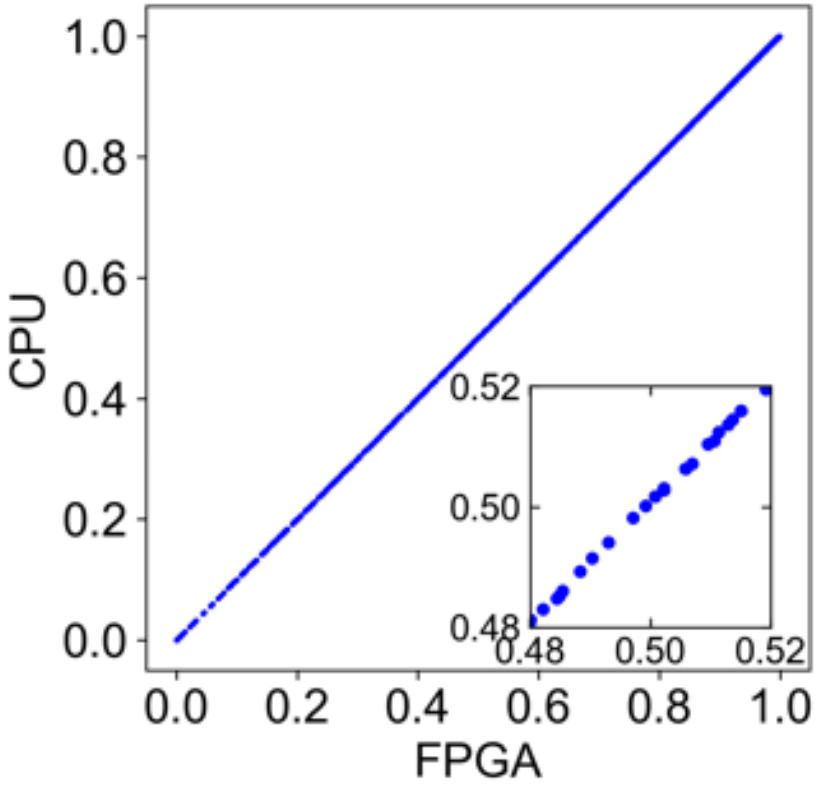


Suzuki *et al*. *Sci. Rep*. **13**, 7735 (2023)

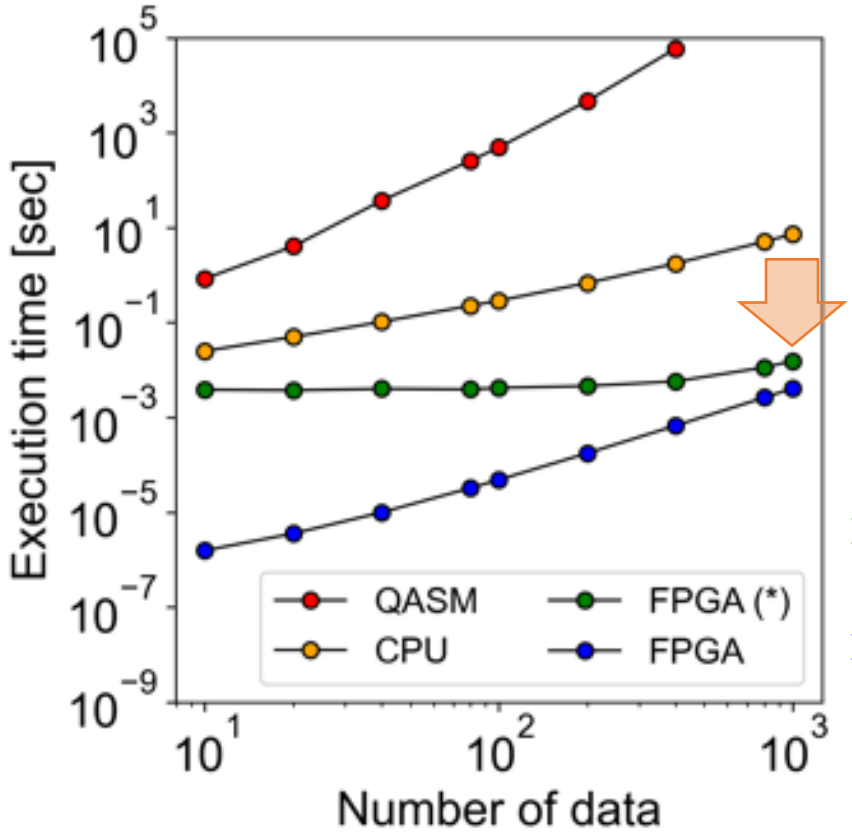
FPGAによる高速化を検証

CPUベースとFPGAアクセラレータで量子カーネル演算を実行した場合の処理速度を比較
FPGAの実行速度は、通信オーバーヘッドを含めても、CPUの実行速度に比べ470倍高速化
CPU/FPGAの量子カーネル値の誤差は± 0.095 %以内に収束

FPGAの精度検証



データ数に対する実行時間



※緑：ホストとの
通信時間込

※青：FPGAカーネル
単体処理時間

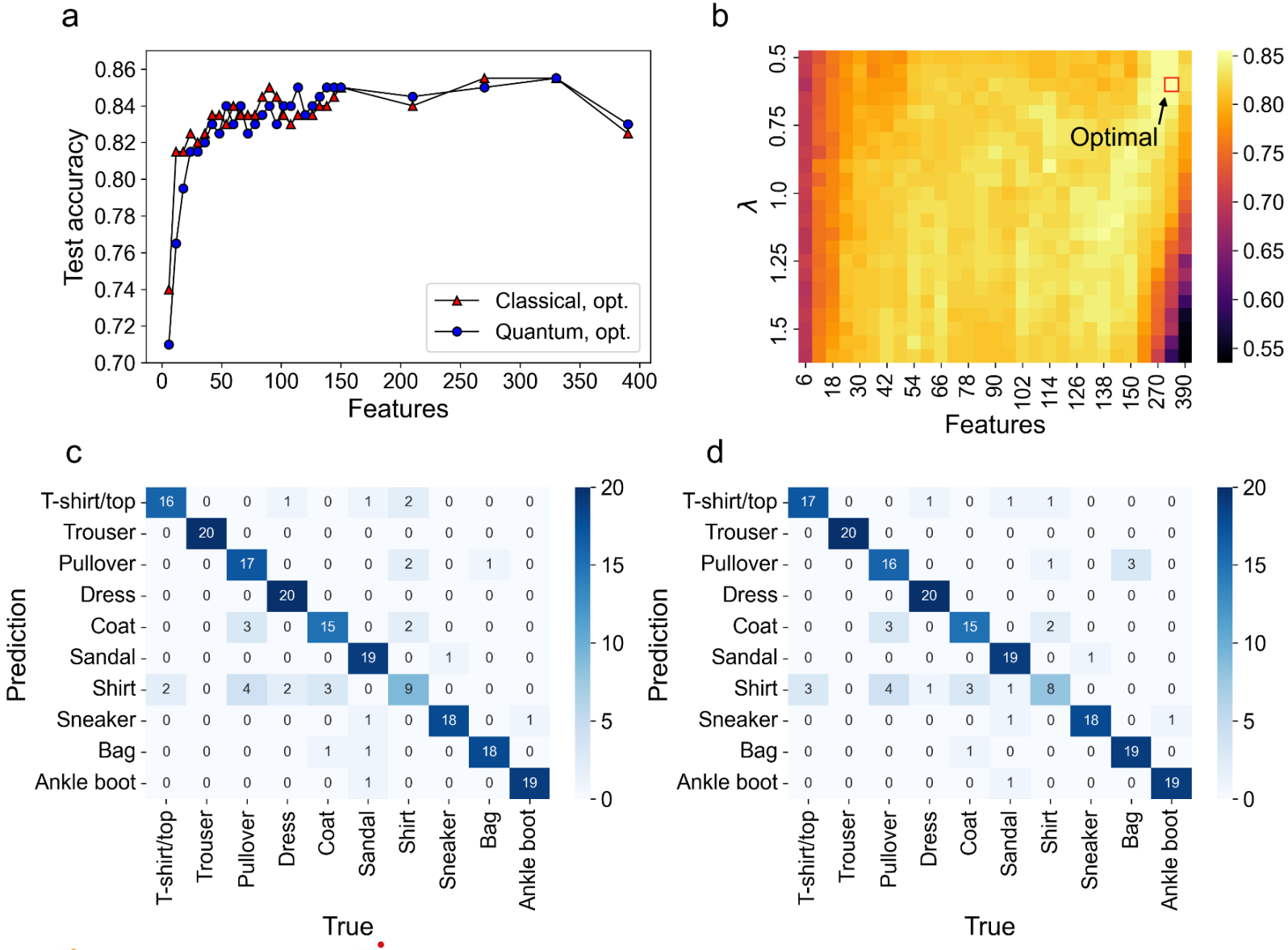
画像データ（Fashion MNIST）の多クラス分類への応用

400次元の特徴に対応可能（a）

予測精度は、古典ガウス
カーネルから得られたモデルと
同等の精度（c, d）

スケーリングパラメータの
チューニングも重要（b）

$$\lambda \ (\text{i.e., } \boldsymbol{x}^{(i)} \leftarrow \lambda \boldsymbol{x}^{(i)})$$



Suzuki *et al*. *Sci. Rep*. **13**, 7735 (2023)

3. FPGA実装

1. 研究背景と概要

2. 量子AIシミュレーター

3. FPGA実装

4. 量子SDKと実機検証

5. まとめ

なぜ、FPGAを使うと高速化するのか？

FPGA（Field Programmable Gate Array）とは、ユーザが論理回路を自由に書き換え可能な集積回路。
特定の計算負荷の高いタスクを論理回路にすることで高速かつ省電力に演算処理を行うことができる。

カスタマイズ性

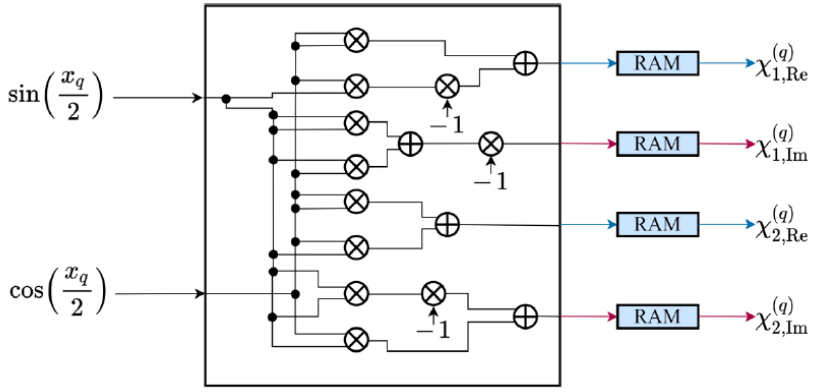
Verilogなどのハードウェア記述言語を使用して、特定のタスクを実行するようにカスタマイズすることができる

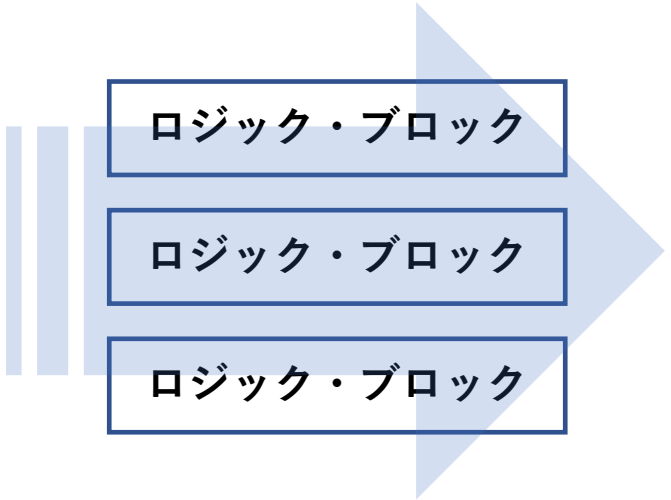
並列処理

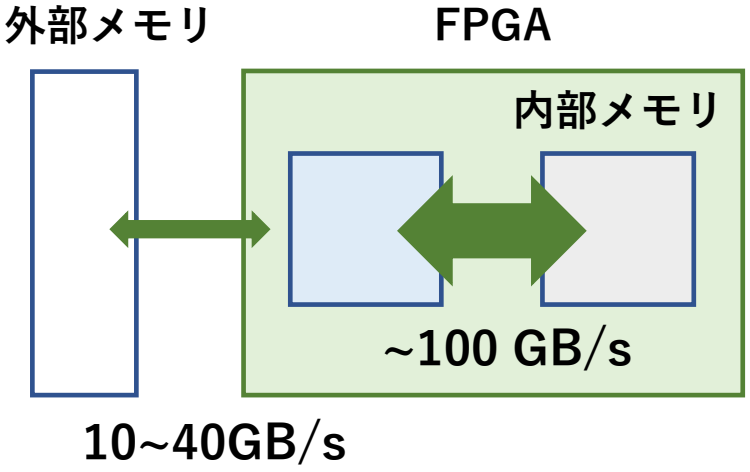
各ロジック・ブロックが効率的に特定のタスクを同時に実行できる

内部メモリ

外部メモリ（オフチップメモリ）にアクセスせず、FPGAハードウェアの内部メモリにアクセスできる

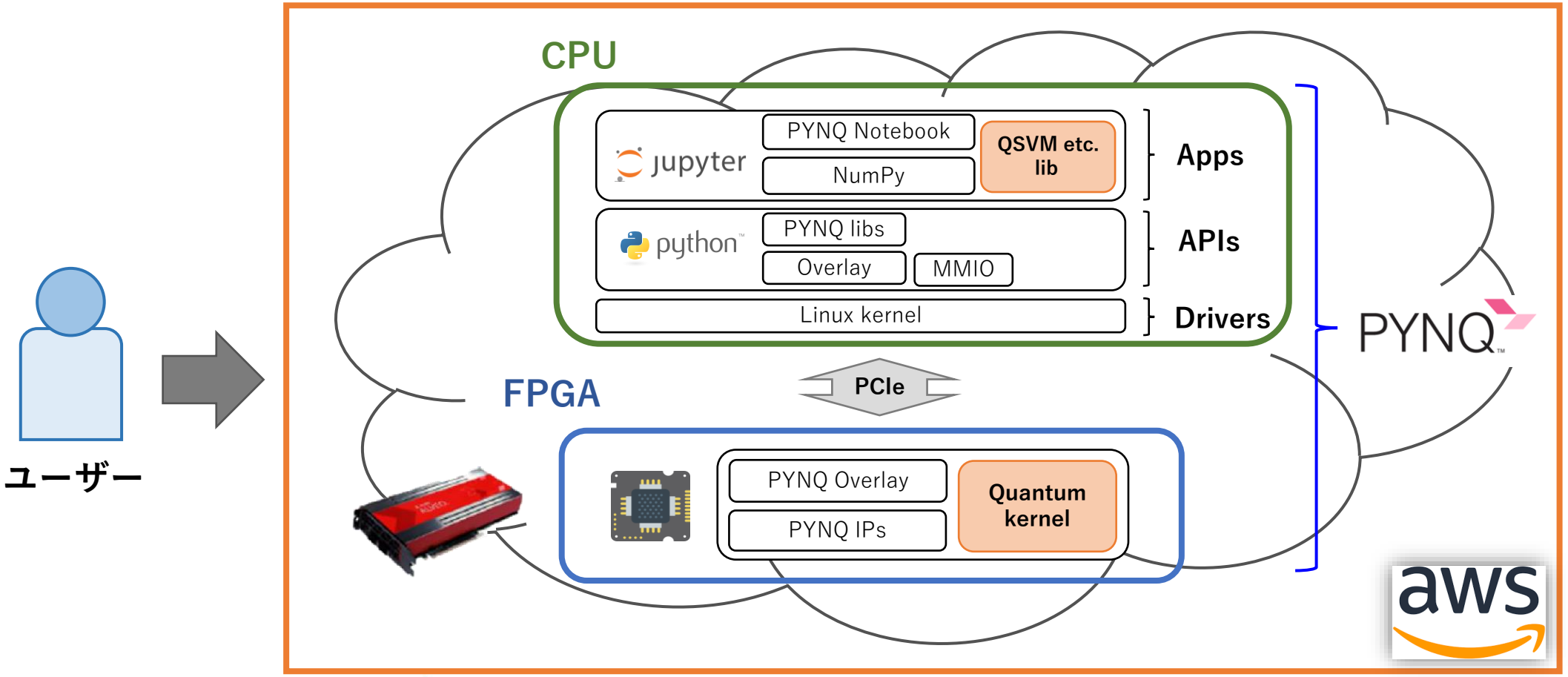






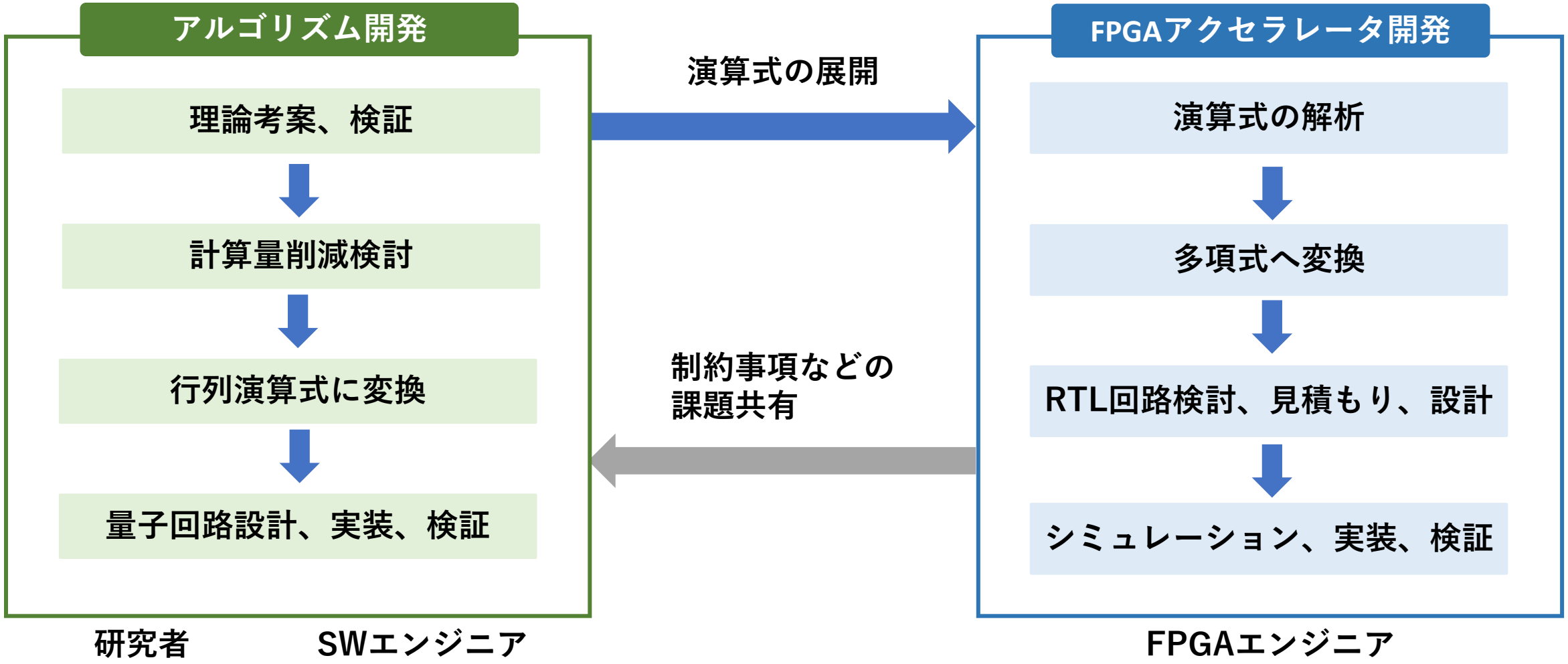
PYNQを活用したFPGAアクセラレータの開発

- PYNQを活用することで、Python + Jupiter Notebookをフロントエンドとして扱えるため
  インタラクティブなFPGAアクセラレータの開発が可能となった
- FPGAに馴染みのないアルゴリズム研究者やソフトウェアエンジニアとの協調もし易くなった



FPGAアクセラレーション開発における協調設計プロセス

アルゴリズム研究者、ソフトウェアエンジニア、FPGAエンジニアの知見を幅広く補完する必要がある



今回のFPGA実装の基本的なアプローチ

今回は、右図の量子回路に固定する
（事前の調査でこの量子回路を使って
MNISTの分類ができることを確認）

▼

行列ベクトル演算に変換する
（一般にユニタリー行列の要素は複素
数）

▼

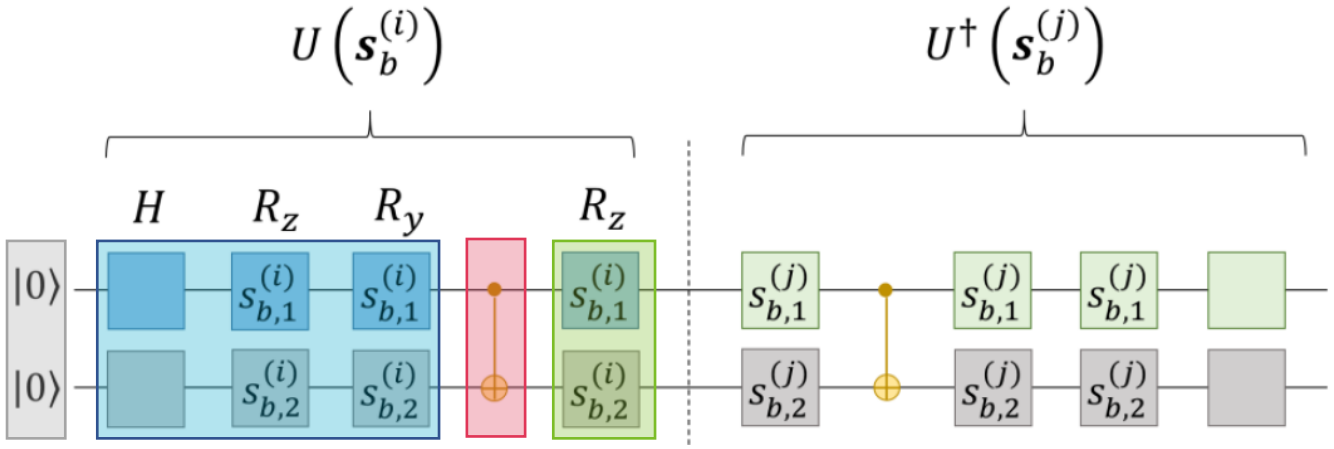
零要素を多く含む行列やベクトルがあり、
アルゴリズム的な工夫により計算量を
削減（詳細は前掲論文を参照のこと）

▼

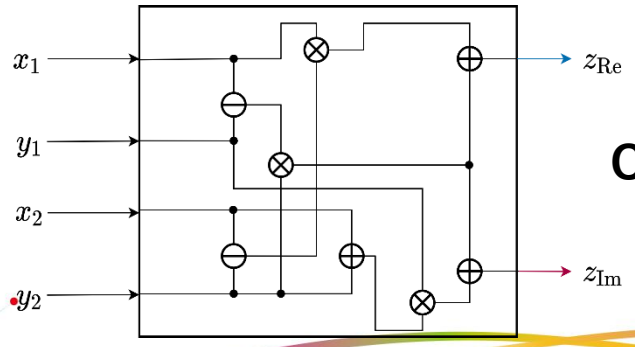
FPGA内での三角関数の計算は、
CORDIC IPを採用し、演算はすべて
16ビット固定小数点演算を採用した
（カーネル行列の要素は、0から1の範囲
に収まるため、ほぼ問題が起こらない）

▼

複素乗算器の総数を見積もり、FPGA
で実行できることを事前に机上検討



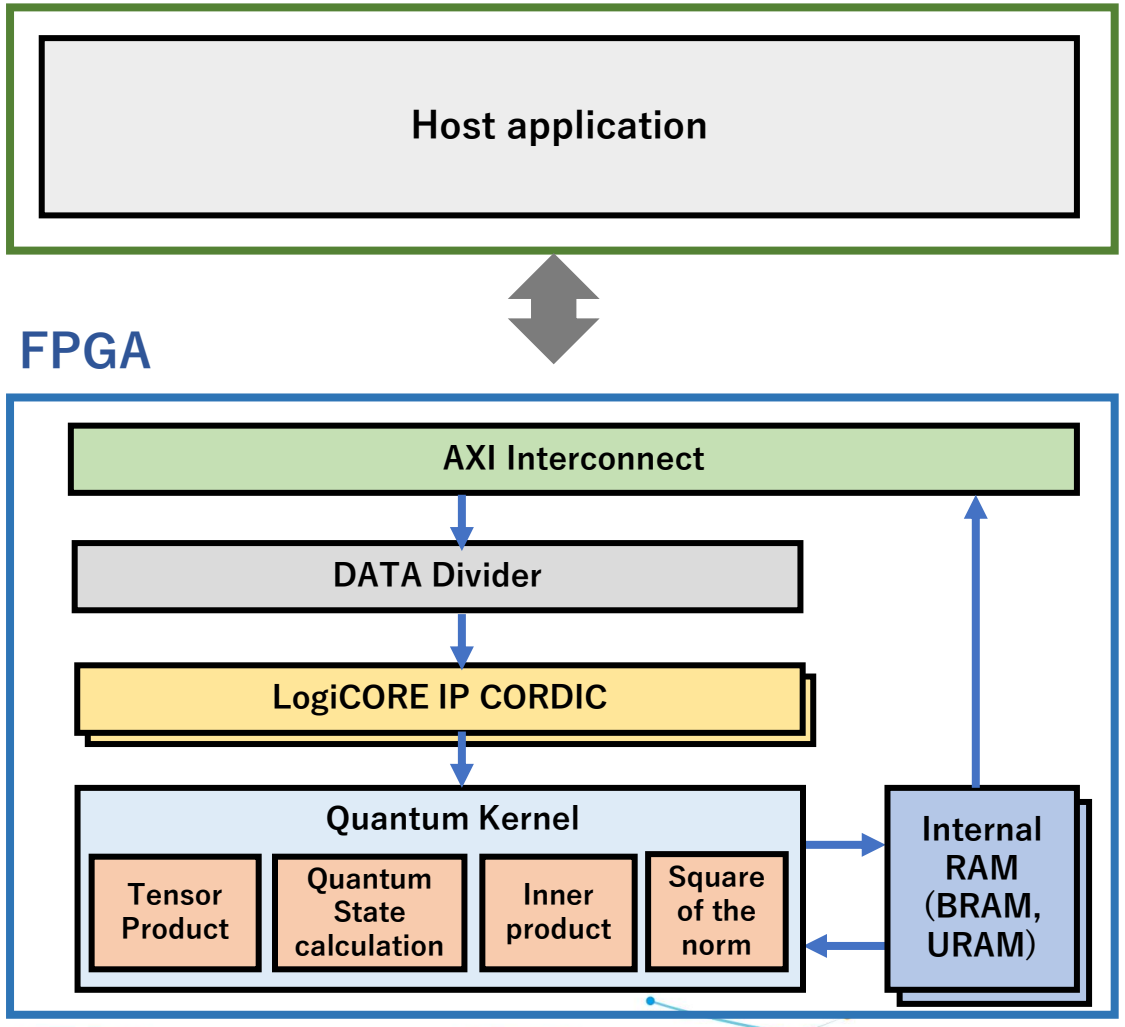
$$\begin{bmatrix} V_{11} & 0 & 0 & 0 \\ 0 & V_{22} & 0 & 0 \\ 0 & 0 & V_{33} & 0 \\ 0 & 0 & 0 & V_{44} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} & U_{13} & U_{14} \\ U_{21} & U_{22} & U_{23} & U_{24} \\ U_{31} & U_{32} & U_{33} & U_{34} \\ U_{41} & U_{42} & U_{43} & U_{44} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



Complex Multiplier

FPGAアクセラレータ開発：全体像

アクセラレーションシステム系統図



アクセラレーションシステム開発フロー

VIVADO

- RTL（Register Transfer Level）設計
- シミュレーション

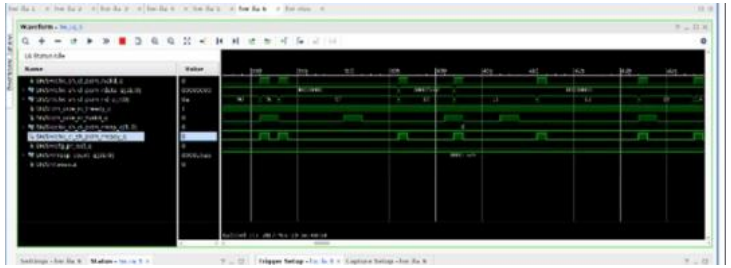
.xo

XILINX VITIS

- ビルド
- インテグレーションレポート確認

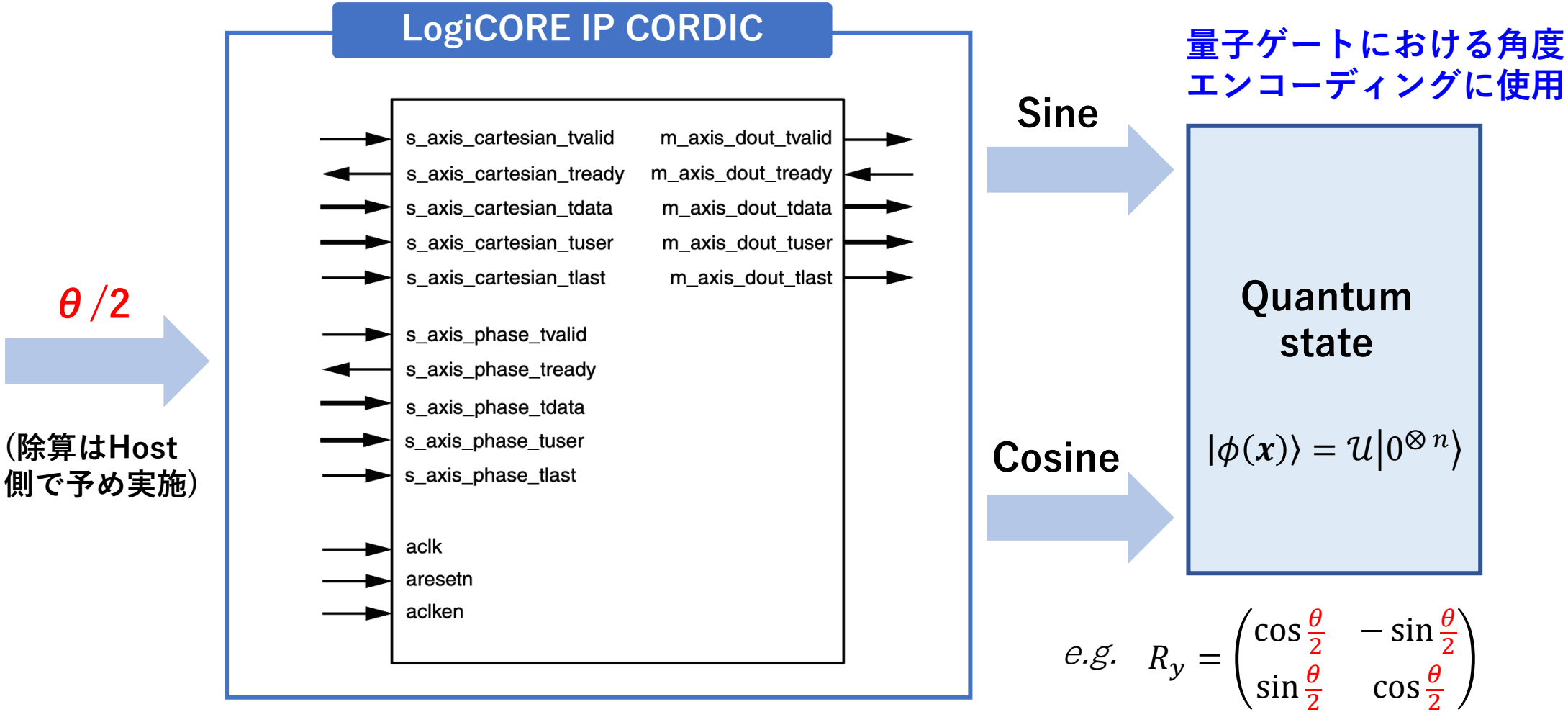
.xclbin
.AFI変換

aws



FPGAアクセラレータ開発：標準IPコアの利用

- 三角関数変換は標準で利用可能なLogiCORE IP CORDIC™ (AMD Xilinx)を利用
- 量子ゲートにおける角度エンコーディングに使用、開発工数削減にも貢献



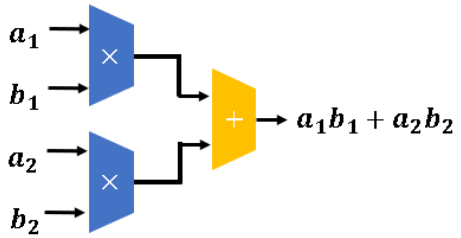
FPGAアクセラレータ開発：メイン演算ユニットの構築

- 行列演算等を多項式形式に変換し、回路設計を実施
- メインのカーネル設計において高位合成は用いず、ほぼ全てRTL記述で設計を実施

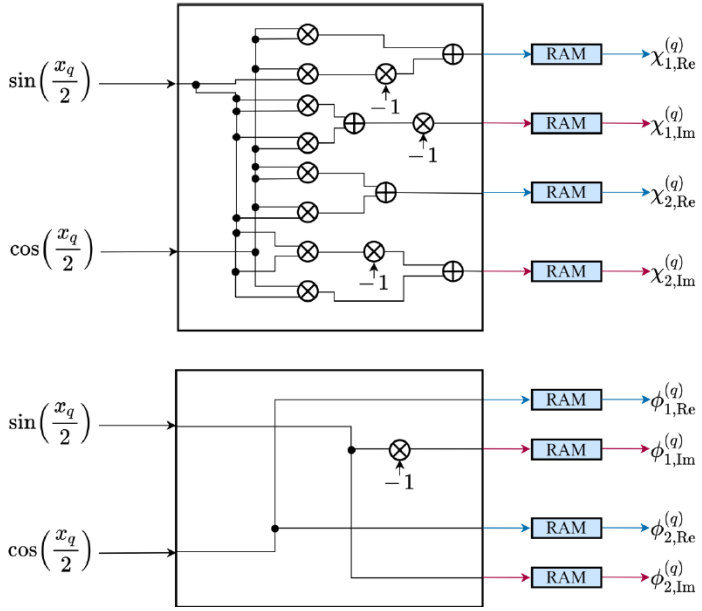
例：ベクトルの内積 ⇒ 多項式変換

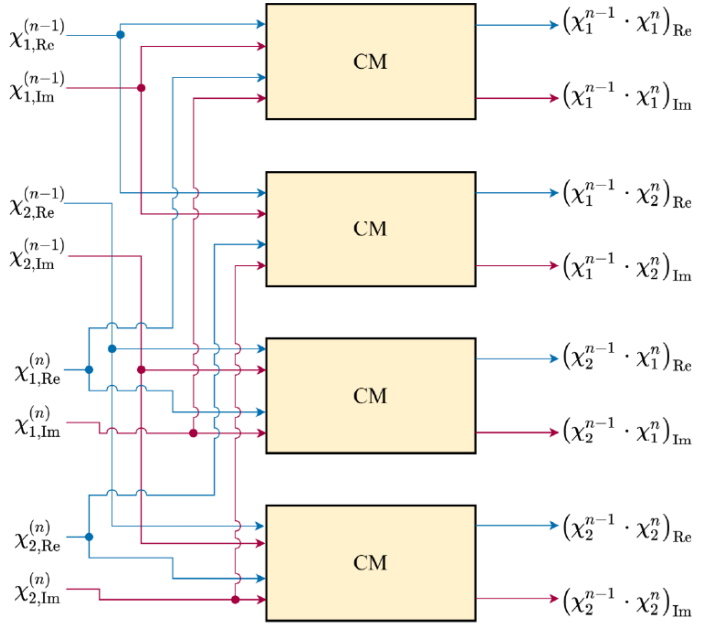
$$\vec{A} = (a_1, a_2), \vec{B} = (b_1, b_2)$$

$$\vec{A} \cdot \vec{B} = a_1 b_1 + a_2 b_2$$



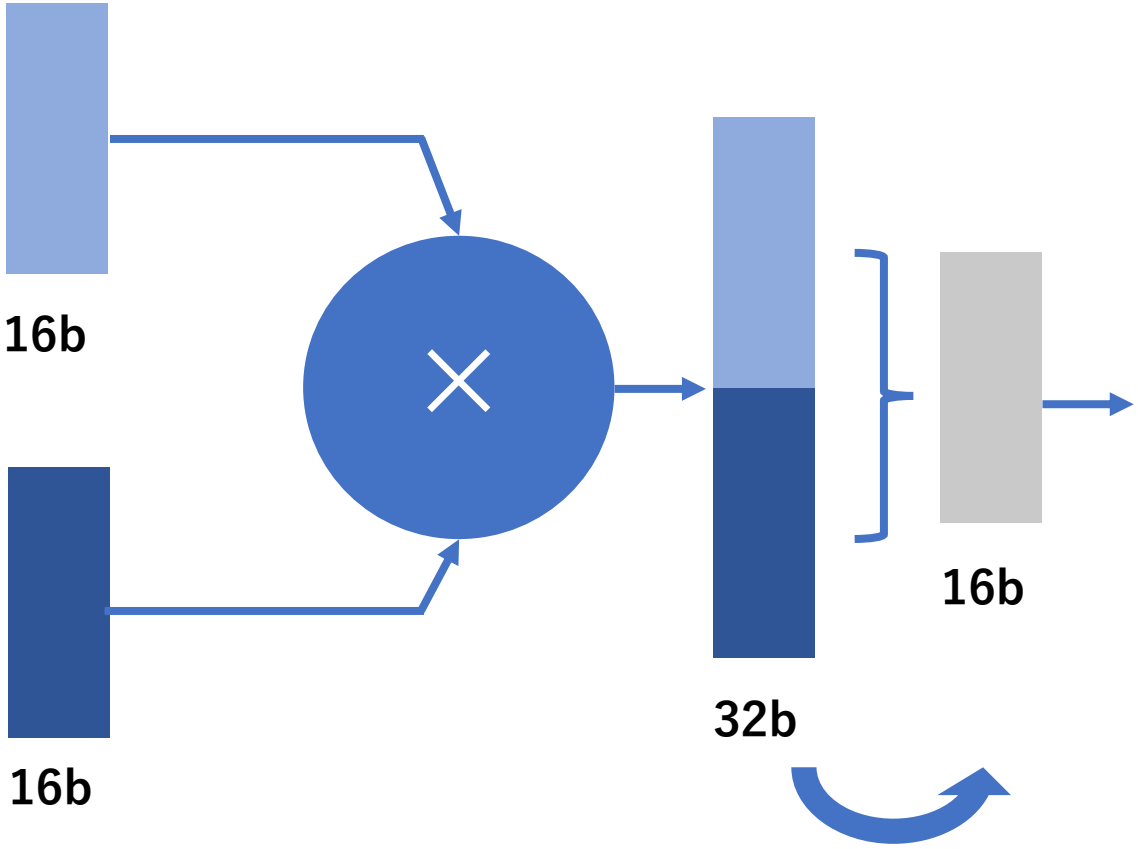
各演算ユニットの並列化などの柔軟性を活かすことで処理の高速化に寄与できる





FPGAアクセラレータ開発：レベルダイヤ検討（乗算器の削減）

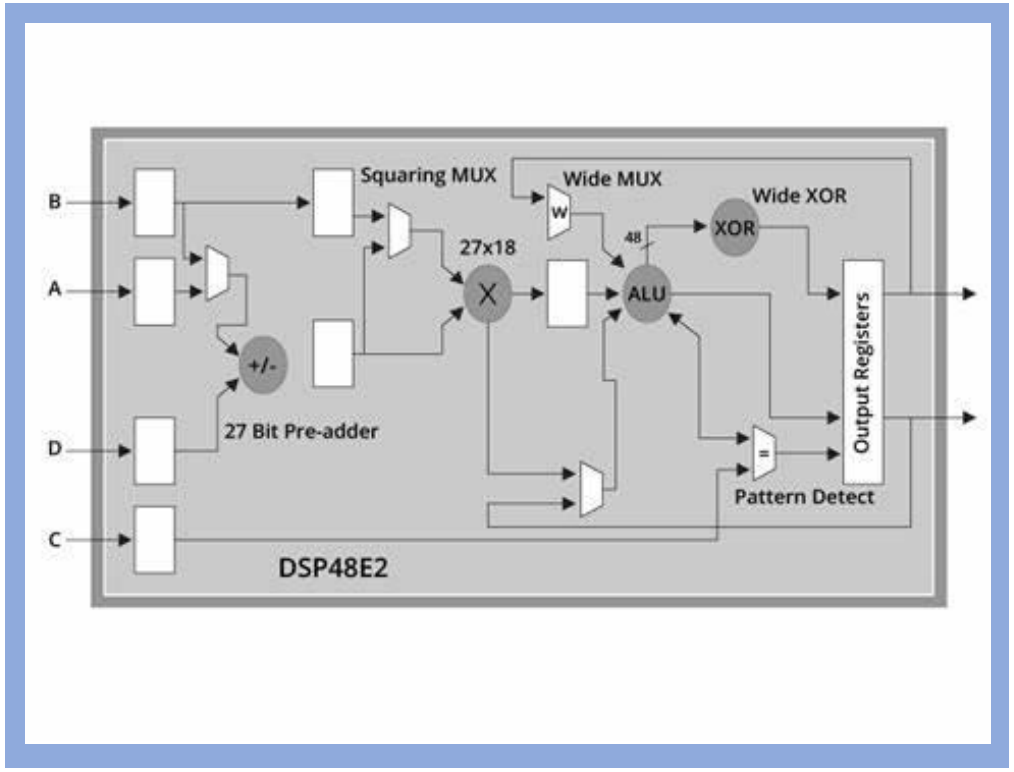
- 一つの演算器に消費されるDSPスライスは2個以上にならないように考慮
- そのため、各演算器におけるビット幅が27×18bitを超えないようなレベルダイヤ設計の検討が必要



固定小数点

精度影響を考慮しながら
レベルダイヤを決定

Xilinx DSPスライス



演算ビット数に応じて相応の
演算ユニットが必要になる

FPGA実装の制限：量子もつれは何量子ビットまで実装可能か

AWS-F１環境において
以下の条件で実装使用率
の比較を行った

①量子ビット(2 qubits)
②量子ビット(6 qubits)

Supplementary Table 1: Hardware utilizations for the quantum kernel implementation

Chip	XCVU9P (AWS F1 instance)	XCVU9P (AWS F1 instance)
Number of qubits	2	6
Train size	1024	1024
LUT	175077/1180984 (15%)	237681/1180984 (20%)
LUTRAM	17335/591440 (3%)	57360/591440 (10%)
FF	250576/2364480 (11%)	327521/2364480 (14%)
BRAM	365.5/2160 (17%)	515/2160 (24%)
URAM	43/960 (4%)	43/960 (4%)
DSP	61/6840 (1%)	1154/6840 (17%)
Clock frequency (MHz)	250	250

Abbreviations: LUT, look-up table; RAM, random access memory; FF, flip flop; BRAM, block RAM; URAM, ultraRAM; DSP, digital signal processor.

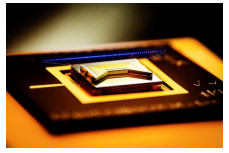
現状、①と②の条件においてLUT/FFやDSP、RAMなどに余裕はあるが、8 qubits以上になるとRAMやDSPの使用率が90%超える見込み。大規模な量子もつれを模擬するために拡大実装するには、相応の工夫が必要。

3. 量子SDKと実機検証

1. 研究背景と概要

2. 量子AIシミュレーター

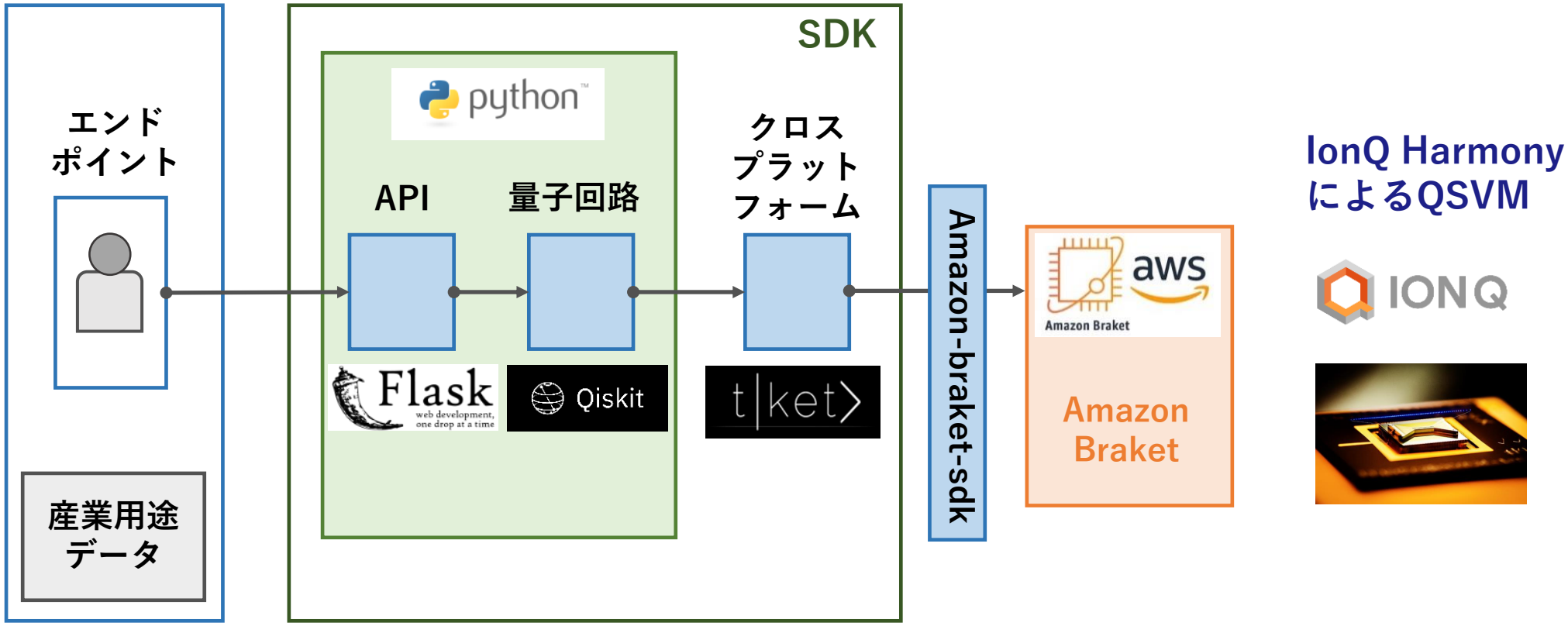
3. FPGA実装



4. 量子SDKと実機検証

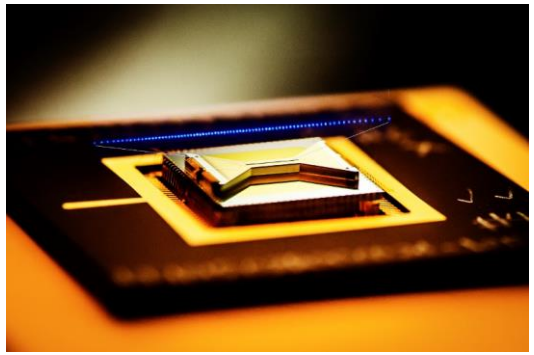
5. まとめ

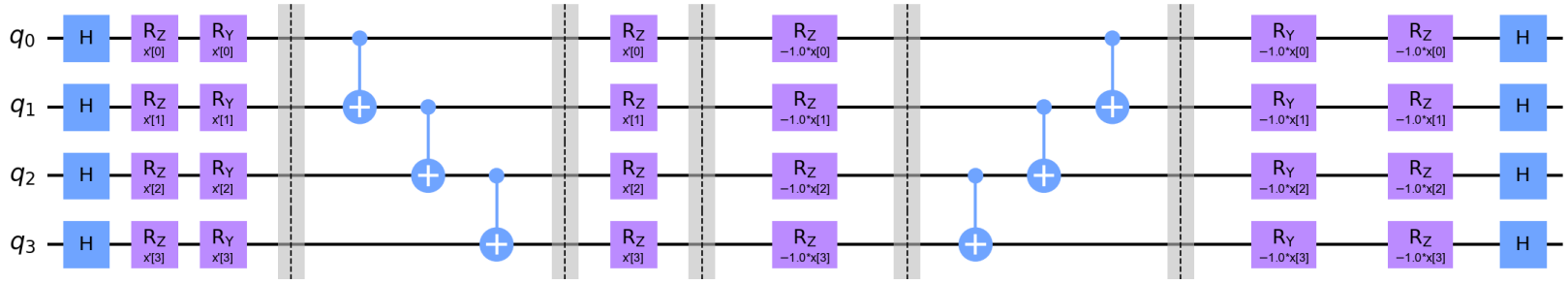
Amazon Braketを利用して、量子カーネル法を実機検証する



今回の量子実験を再現するための量子ソフトウェア開発キット（SDK）は、以下でで入手できる
https://github.com/scsk-quantum/phiqonnect

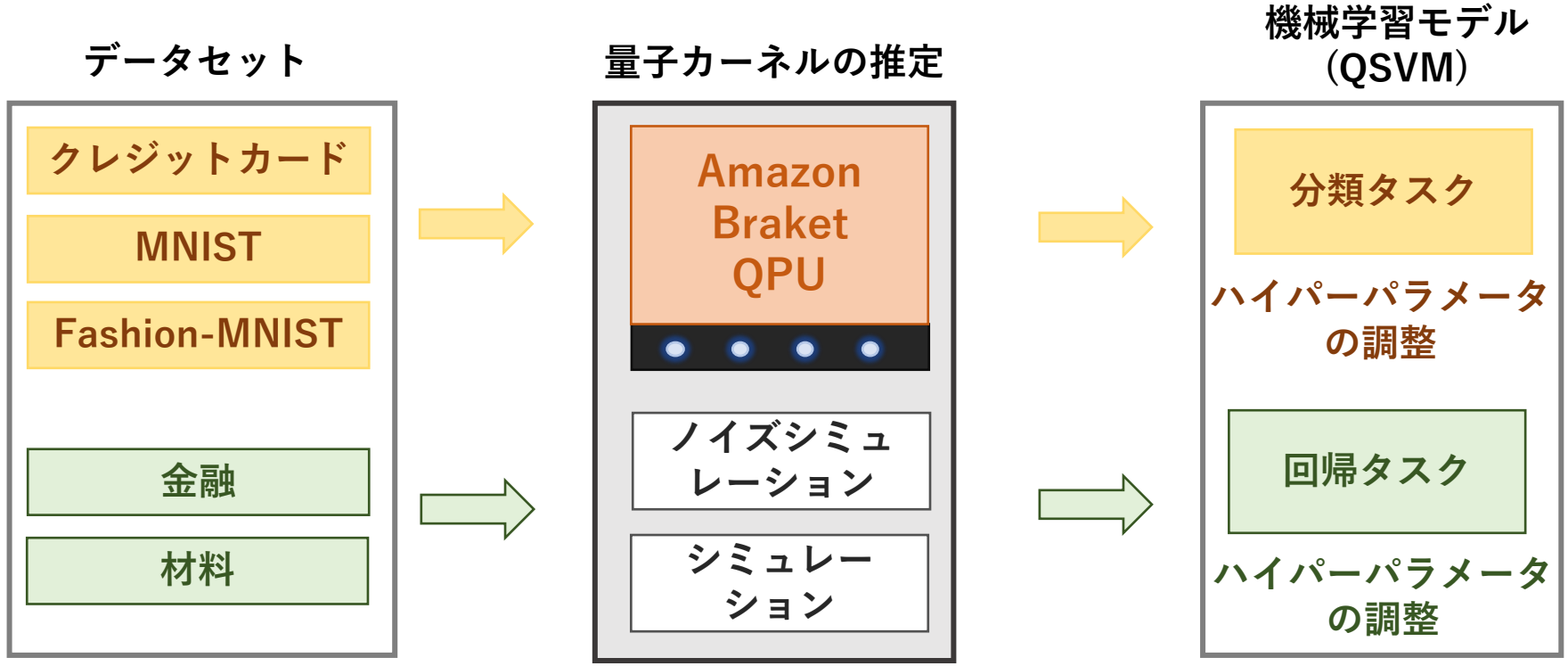
IonQ Harmonyによる量子サポートベクトルマシン（QSVM）





実際のデバイスで実装可能な浅い量子回路を使用

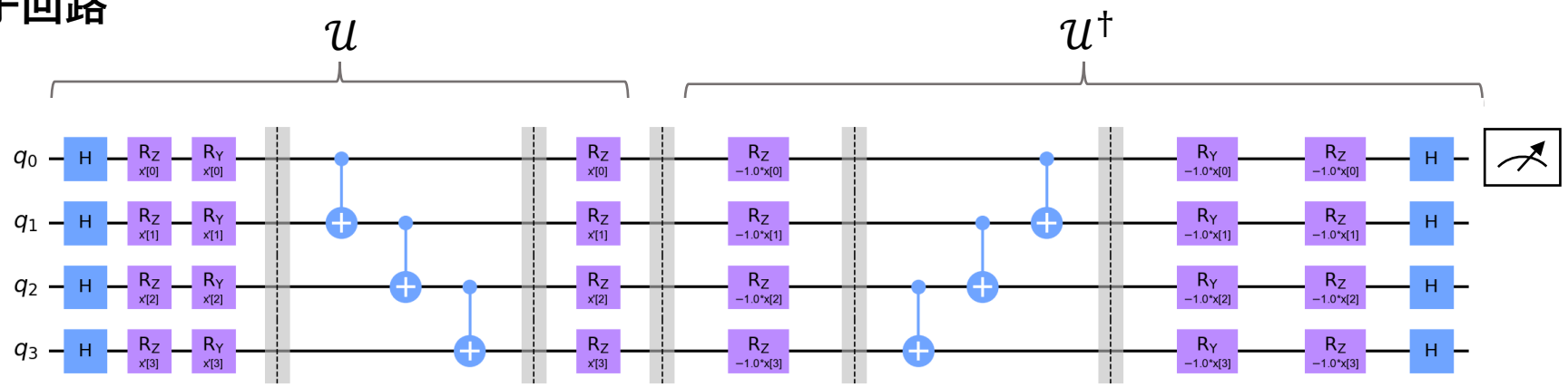
同じ量子回路のテンプレートを5つのデータセットに使用



Suzuki *et al*. arXiv: 2307.02091 (2023)

浅い量子回路による量子サポートベクトルマシン

量子回路



量子特徴写像 §

$$|\phi(\boldsymbol{x})\rangle = \left(\otimes_{q=1}^{n} R_z(x_q)\right) U_{2^n}^{\text{ent}} \left(\otimes_{q=1}^{n} \left(R_y(x_q) R_z(x_q) H\right)\right) |0^{\otimes n}\rangle \quad \textbf{with} \quad U_{2^n}^{\text{ent}} := \prod_{q=1}^{n-1} \mathbf{CNOT}_{q,q+1}$$

量子カーネル

$$K(\boldsymbol{x}, \boldsymbol{x}') = |\langle\phi(\boldsymbol{x})|\phi(\boldsymbol{x}')\rangle|^2 \qquad \text{(NISQ device)}$$

- 最隣接の接続

⇒ $(n-1)$ の接続

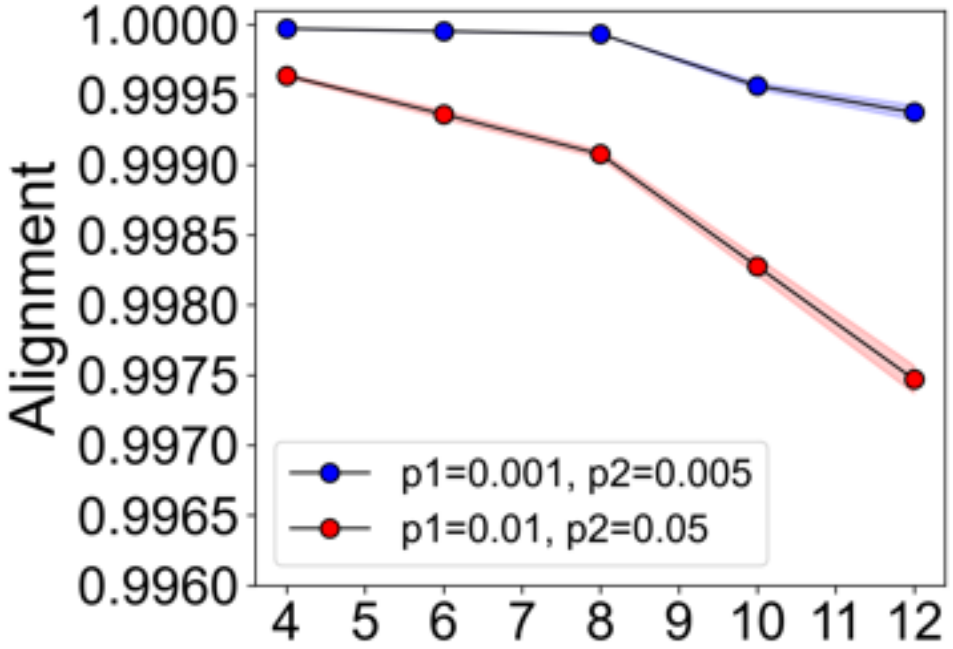
Decision関数

$$f(\boldsymbol{x}) = \text{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_j) + b\right)$$

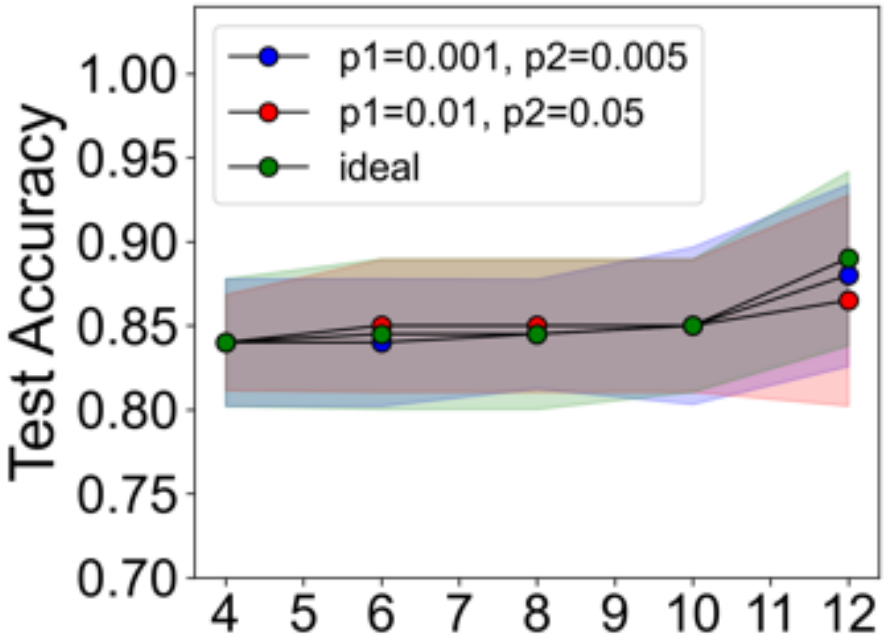
§ FPGAの量子AIシミュレーターでも同じ量子回路を使用している

デバイスノイズ・シミュレーションによる量子ビット数の拡張性

行列のアラインメント



精度（テスデータ）



Credit card data

- アラインメントは高い値を維持

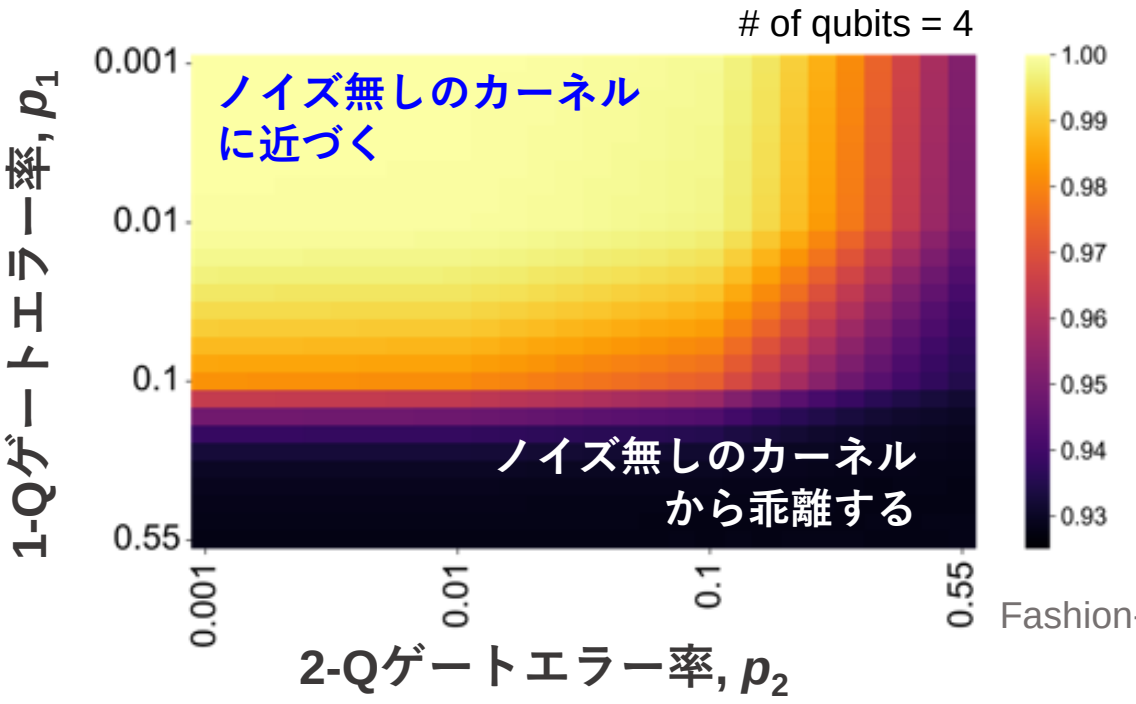
$$A\left(K, K^{\text{noise}}\right) = \frac{\langle K, K^{\text{noise}} \rangle_F}{\sqrt{\langle K, K \rangle_F \langle K^{\text{noise}}, K^{\text{noise}} \rangle_F}}$$

- 量子ビット数を4〜12で増加させた場合でも、精度は維持された

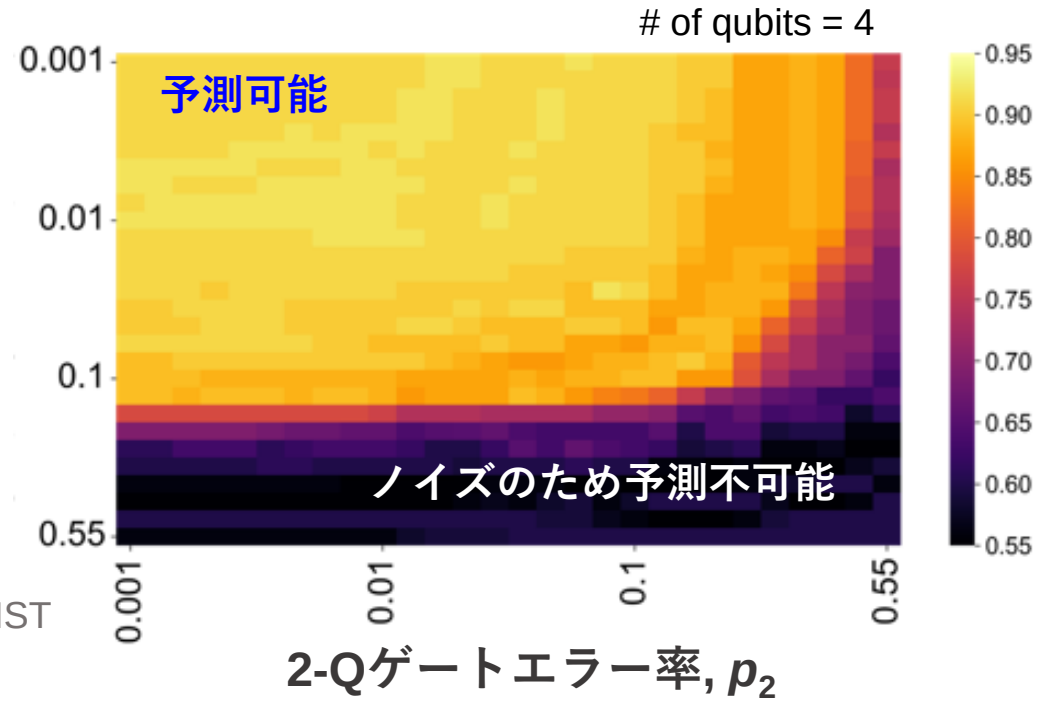
Suzuki *et al*. arXiv: 2307.02091 (2023)

ゲートエラー率の影響：ノイズ環境下でも分類タスクが可能な領域がある

行列のアラインメント



精度（テストデータ）



Fashion-MNIST

$$A\left(K, K^{\text{noise}}\right) = \frac{\langle K, K^{\text{noise}} \rangle_F}{\sqrt{\langle K, K \rangle_F \langle K^{\text{noise}}, K^{\text{noise}} \rangle_F}}$$

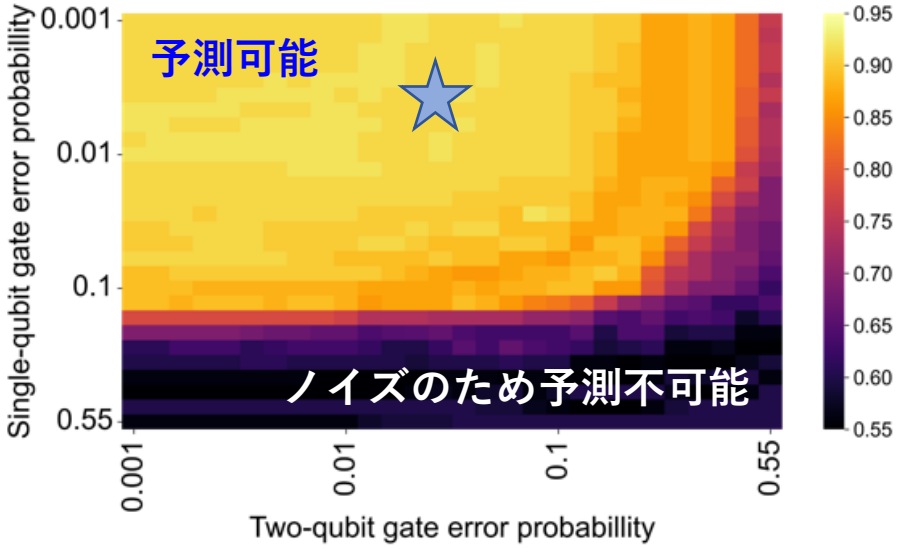
- ある特定の領域（$p_1 < 0.05 \wedge p_2 < 0.1$）において、ノイズの存在下であっても、QSVMモデルは予測することができた

Suzuki *et al*. arXiv: 2307.02091 (2023)

量子サポートベクトルマシン：分類 (QSVC)

4量子ビットを使用したIonQ HarmonyにおけるAIモデルの予測性能は、古典カーネルと同等の精度となった

様々な量子ビットゲートエラーによるデバイスノイズ・シミュレーション



3つの異なるデータセットにおける古典および量子SVMのモデル精度

データセット	モデル	精度 (教師) (%)	精度(テスト) (%)
クレジットカード			
	古典SVM	80	70
	QSVM (ノイズ無)	85	70
	QSVM (**IonQ Harmony**)	100	70
MNIST			
	古典SVM	100	100
	QSVM (ノイズ無)	100	100
	QSVM (**IonQ Harmony**)	100	100
Fashion-MNIST			
	古典SVM	100	100
	QSVM (ノイズ無)	100	100
	QSVM (**IonQ Harmony**)	100	100

c.f. IonQ Harmony:
1量子ビットゲートエラー: 0.004
2量子ビットゲートエラー (ネイティブゲート): 0.027

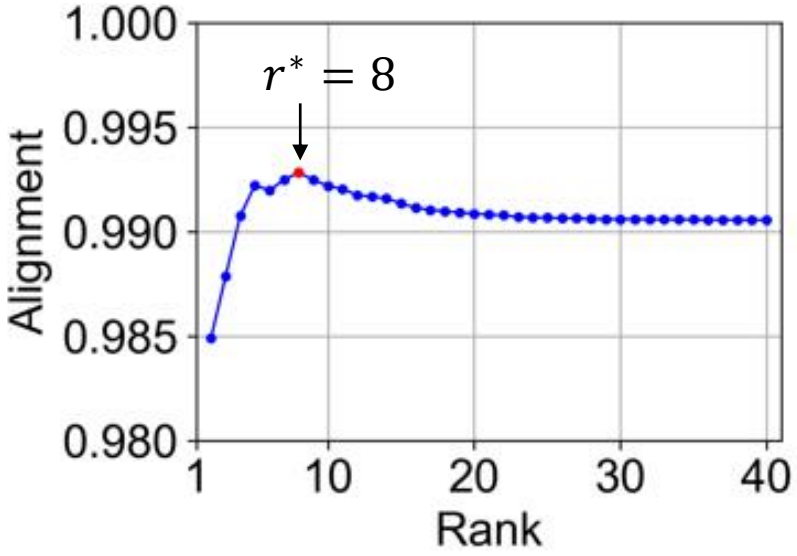
量子カーネルのノイズを軽減する

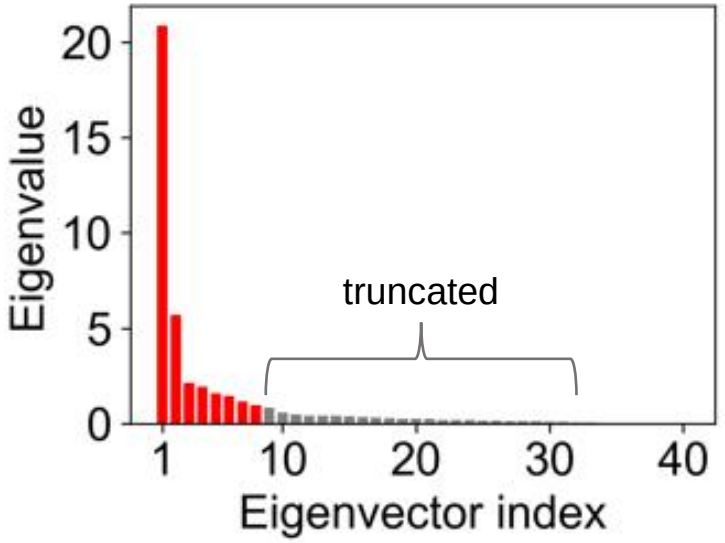
- ノイズのないカーネルとの
  アラインメントを最大化

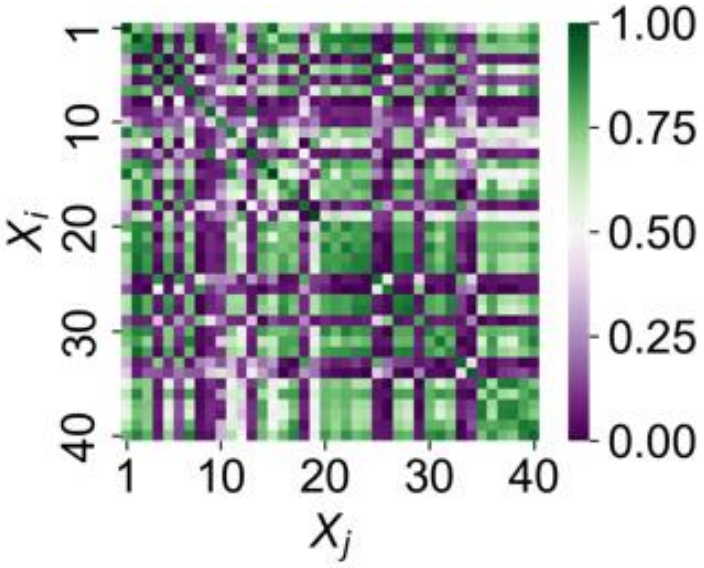
$$r^* = \underset{r \in \mathbb{N}}{\operatorname{argmax}} A\left(K, \widehat{K}\right)$$

$$\widehat{K} = \sum_{k=1}^{r^*} \mu_k \boldsymbol{u}_k \boldsymbol{u}_k^{\mathrm{T}}$$

- ノイズは、低ランク近似（特異値分解）
  によって軽減することができる
- 低ランク近似は、元の行列の本質的な情
  報を捉えることができる







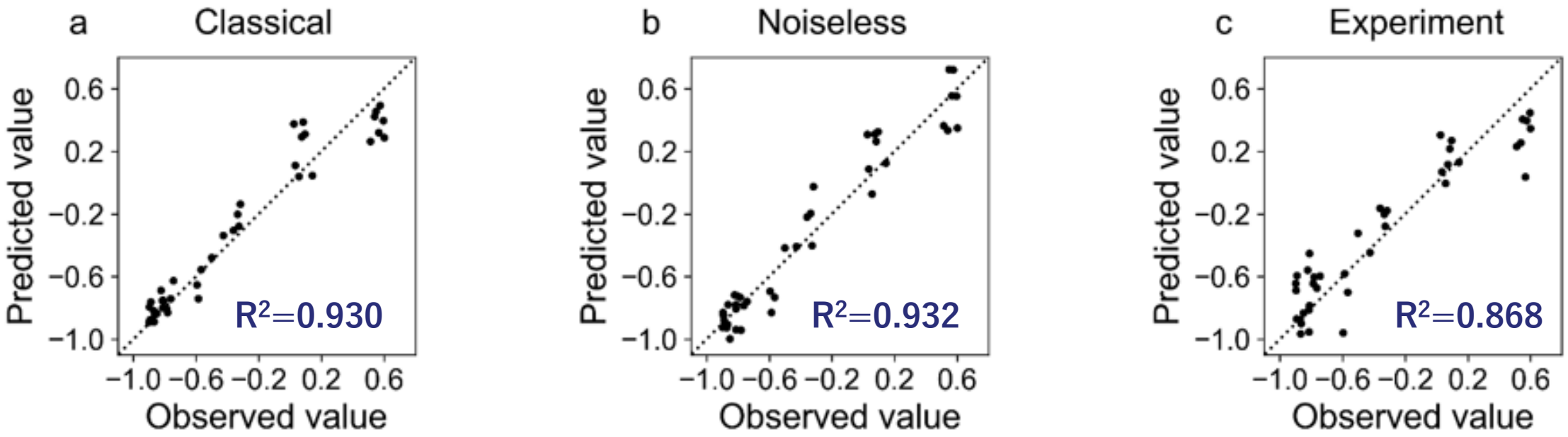
Suzuki *et al*. arXiv: 2307.02091 (2023)

量子サポートベクトルマシン：回帰 (QSVR)

- 検証の結果は、実際のデバイスにおけるQSVRモデルの実行可能性が示唆された
- 我々の量子カーネルは、異なるデータセットに対してQSVCおよびQSCRタスクの両方で取り扱うことができる

Qiskit

IONQ Harmony



Suzuki *et al*. arXiv: 2307.02091 (2023)

浅い量子回路とテンソルネットワーク

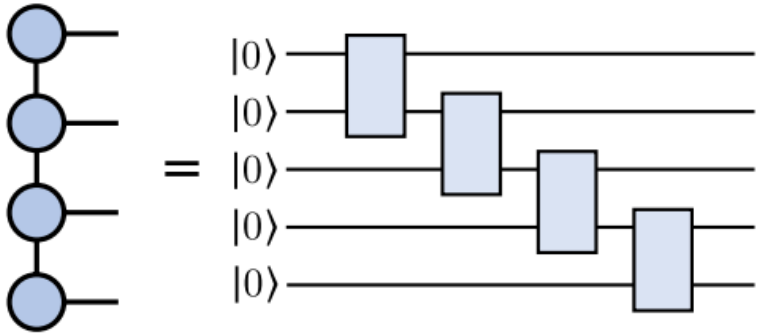
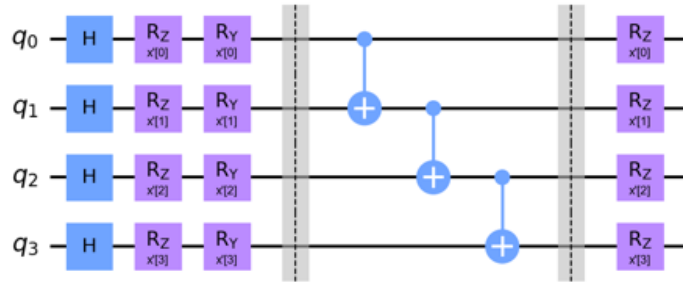


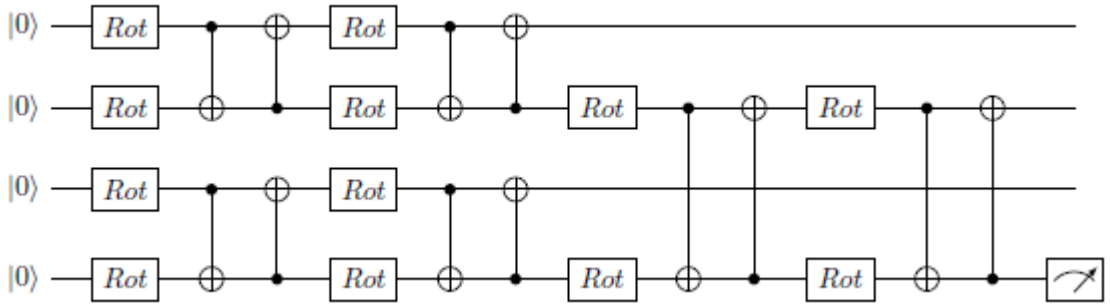
Figure 2. An example of a matrix product state (MPS) (*left*) and the corresponding quantum circuit architecture (*right*). Each tensor in the network is obtained by contracting the initial two-qubit state and a two-qubit gate.

Matrix Product State（MPS）と対応する量子回路



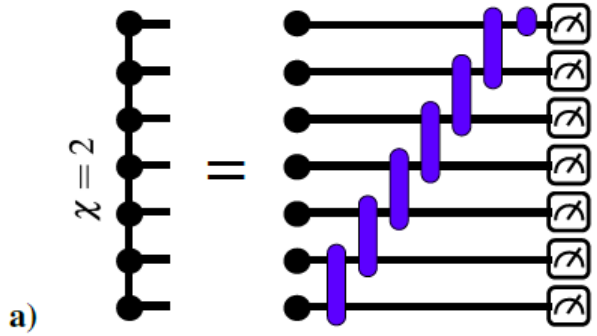
Suzuki *et al*. arXiv: 2307.02091 (2023)

画像分類のためのTree tensor量子回路



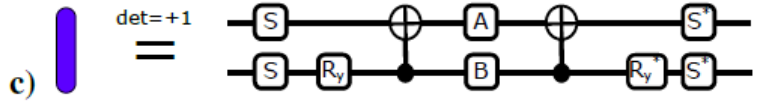
Guala *et al*. *Sci. Rep*. **13**, 4427 (2023)

確率分布を生成するMPS量子回路



MPSを用いて、最大20量子ビットのイオントラップ量子コンピューターで正規確率分布のクラスをエンコードする量子状態を生成

Iaconis *et al*. arXiv:2303.01562 (2023)



量子AIシミュレーター開発のまとめ

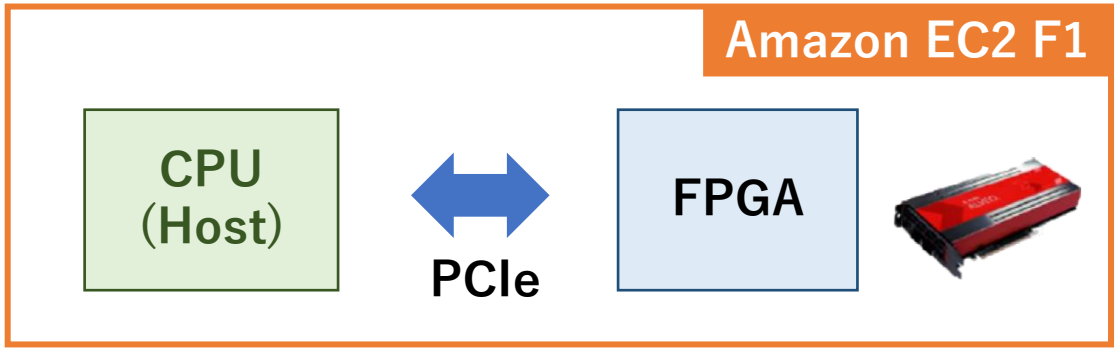
現状の量子コンピュータは制限事項が多々あり、量子技術の研究開発や、社会実装を推進するためにシミュレーション技術の活用が必要

FPGAの様々な柔軟性を活かすことで、高負荷処理の大幅な高速化が実現可能。特に独自開発のアルゴリズム処理最適化を実現する上ではFPGAとの相性が良い

メディア掲載

日刊工業新聞
電子デバイス産業新聞 Electronic Device Industry News
YAHOO! JAPAN ニュース
クラウド Watch
デジタルビジネスを加速する専門情報サイト IT Leaders
dmenu
ニュースイッチ NEWSWITCH
MONOist

Amazon EC2 F1 instances (FPGA)



量子AIシミュレーター

- CPUベースの手法より470倍高速化できた
- 浅い回路を使って古典カーネルと同等の精度を達成した

Suzuki *et al*. *Sci. Rep*. **13**, 7735 (2023)

謝辞：開発にあたり、ご支援いただいたAMD Xilinx社の住川様、安藤様にお礼申し上げます

量子SDKによる実機検証のまとめ

本研究は、理論的な研究と量子コンピューティングの実応用とのギャップを埋めるべく、量子時代に準備することを目的とした

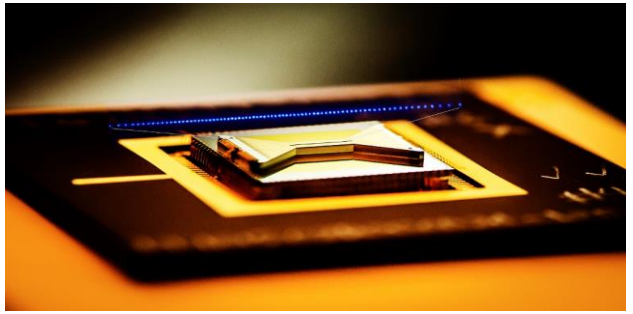
量子カーネル法に焦点を当て、ノイズのある/なしの量子回路シミュレータ（Qiskit）および実機IonQ Harmonyを利用して、QSVCおよびQSVRモデルを構築した

これらのタスクには、不正なクレジットカード取引や、金融データなどの様々なオープンデータセットを使用した

量子カーネル法に特化した量子ソフトウェア開発キット（SDK）を使用し、4量子ビットのイオントラップ型量子コンピュータ上でのQSVCモデルとQSVRモデルの実行可能性を示した

今後は、IonQの次世代機での検証を検討している

Amazon Braket (QPU)



IonQ HarmonyにおけるQSVM

- 4量子ビットを使ったQSVCモデルは、エラー軽減法無しで、ノイズのある環境下であっても信頼性のある予測を行うことを実証した

Suzuki *et al*. arXiv: 2307.02091 (2023)

SDK: https://github.com/scsk-quantum/phiqonnect