Compilation of Entangling Operations for High-Dimensional Quantum Systems

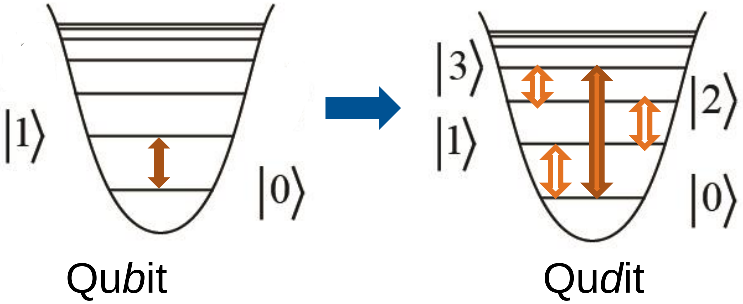
Kevin Mato[1], Martin Ringbauer[2], Stefan Hillmich[3], and Robert Wille[1,4]

[1]Technical University of Munich, Munich, Germany
[2]University of Innsbruck, Innsbruck, Austria
[3]Johannes Kepler University Linz, Austria
[4]Software Competence Center Hagenberg (SCCH) GmbH, Austria

kevin.mato@tum.de
Technical University of Munich
https://www.cda.cit.tum.de/research/quantum

Introduction: Qudits

*



- *Subspace operations*
- *Global Hilbert space operations*

Why a compiler?

- Qudits can be implemented on the latest quantum technologies
- Mixed Dimensional Systems

- Much richer entanglement structure of qudits compared to qubits
- Better circuit complexity and algorithmic efficiency, at an increasing design cost

*https://labsites.rochester.edu/bloklab/index.php/qudits/

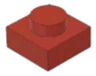
Entanglement Structures

- Much richer entanglement structure of qu*d*its compared to qubits

*

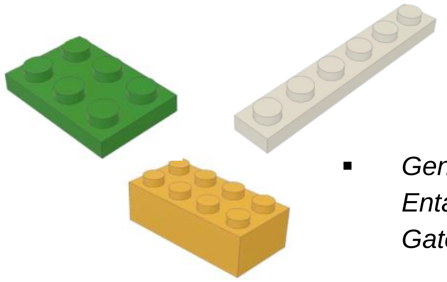

- Quantum Algorithm or Functionality

- *Local Operations*
- *Entangling Operations*



- *CNOT.*

+



- *Genuine Entangling Gates.*

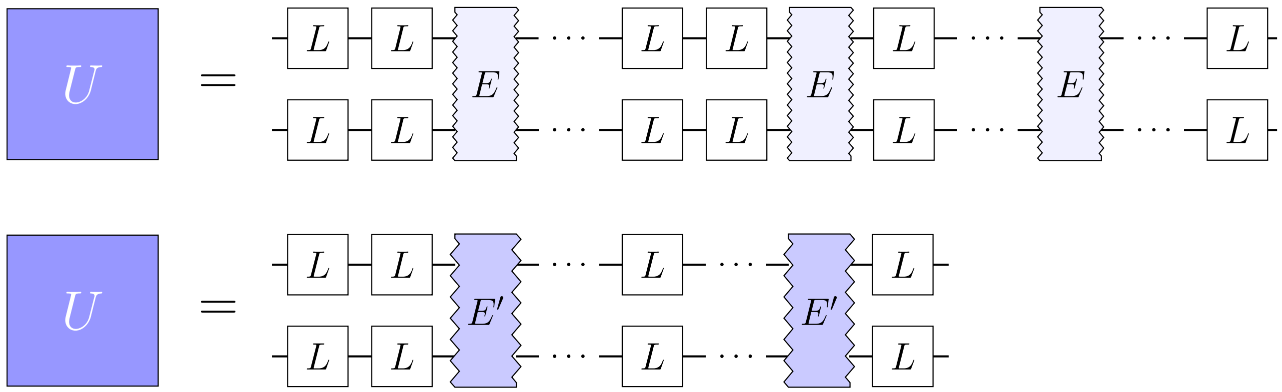
- Challenges in finding suitable gate sets native to hardware and compilation algorithms for these gate sets
- Theory and design methods are insufficient, therefore qudit compilation is still manual

- Once you are given an *unknown arbitrary two-qudit unitary* it is **not** possible to understand beforehand if it is entangling, without performing expensive computations or experiments

- How can you efficiently implement an arbitrary two-qudit unitary given the native gate set of the device?

*https://rebrickable.com/mocs/MOC-110625/Simon%20Lenz/back-to-the-future-time-machine-mods-improvements/#details

Problem

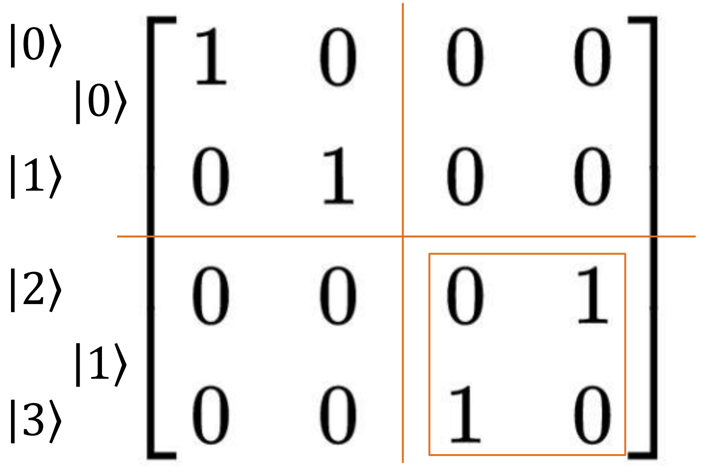


- Given a unitary U representing an interaction between two qudits of dimension d
  Find a decomposition of U into arbitrary local unitaries and a pre-defined set of entangling gates
  In a way that is as close to the optimum as possible.

- Each decomposition takes into account the structure of the entangling gate provided by the quantum hardware and the cost of each gate

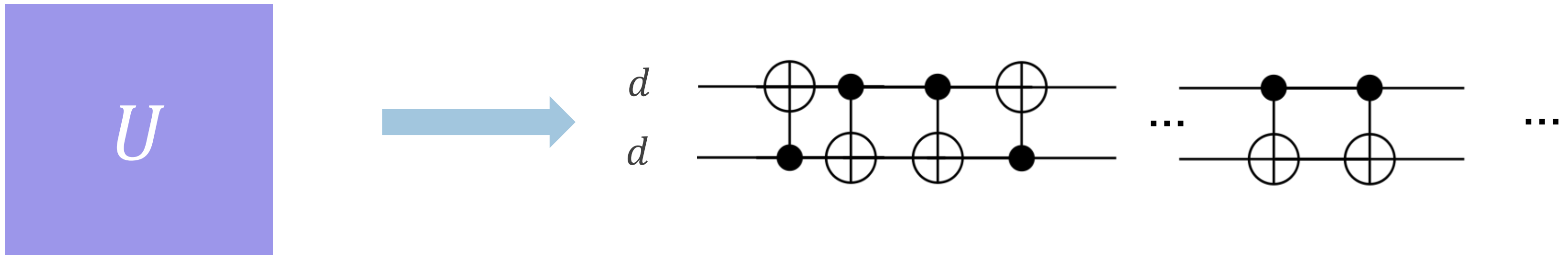
- A compilation workflow in ***2 steps***

Decomposition: First Step

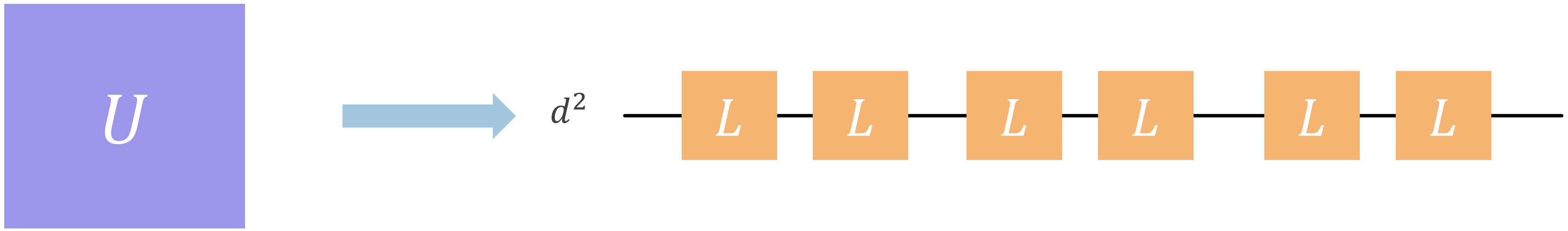


- One of the advantages of using qudits:
  Trading entangling operations for local ones

- We need to quantify the entangling interactions between the two qudits



- We map the target unitary on two-qudits, to an appropriate single qudit unitary, by re-encoding



QR Decomposition

- Given Unitary U find decomposition:

$$U = V_k \cdot V_{k-1} \cdots V_1 \cdot \Theta$$

- Two-level Rotations
- Arbitrary Phases

$$U = V_3 \cdot V_2 \cdot V_1 \cdot \Theta_2 \cdot \Theta_1$$

Initial Unitary:

$$\begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & \frac{e^{\frac{2 i \pi}{3}}}{\sqrt{3}} & \frac{e^{-\frac{2 i \pi}{3}}}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & \frac{e^{-\frac{2 i \pi}{3}}}{\sqrt{3}} & \frac{e^{\frac{2 i \pi}{3}}}{\sqrt{3}} \end{pmatrix} \rightarrow \begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \sqrt{\frac{2}{3}} & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} \\ 0 & -\frac{i}{\sqrt{2}} & \frac{i}{\sqrt{2}} \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & -\frac{i}{\sqrt{2}} & \frac{i}{\sqrt{2}} \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & i \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & i \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- The fixed sequence is a ready-to-use tool
- The QR decomposition creates an overhead

Rotations – from Local to Entangling

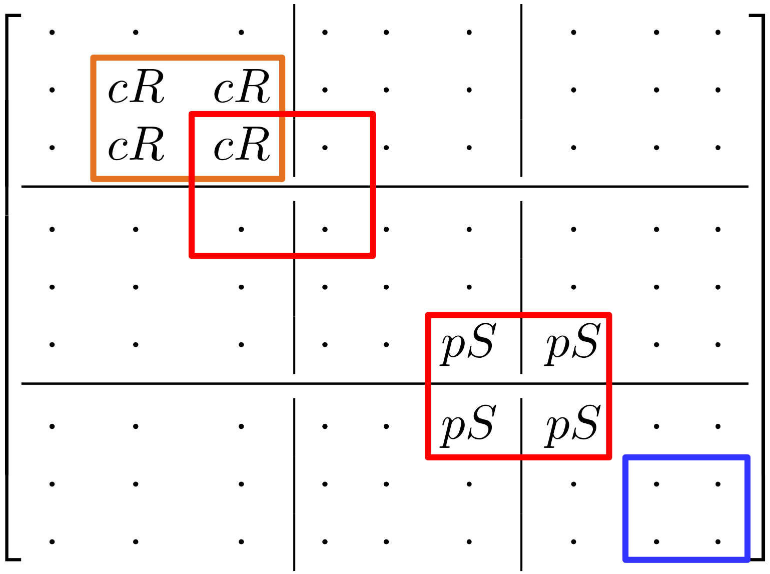
- The result is a sequence of two-level rotations, or *Givens* rotations

$$R(\theta,\phi) = \begin{bmatrix} \cos\frac{\theta}{2} & \sin\frac{\theta}{2}(-i\cos\phi - \sin\phi) \\ \sin\frac{\theta}{2}(-i\cos\phi + \sin\phi) & \cos\frac{\theta}{2} \end{bmatrix}$$

- Due to the choice of encoding, the rotations occur between adjacent states, and are embedded in a d^2 Hilbert space

$$\hat{R}_i(\theta,\phi) = \begin{bmatrix} 1 & \cdots & 0 \\ 0 & \cdots [R(\theta,\phi)]_{i,i+1} \cdots & 0 \\ 0 & \cdots & 1 \end{bmatrix}$$

- There are different types of rotations:

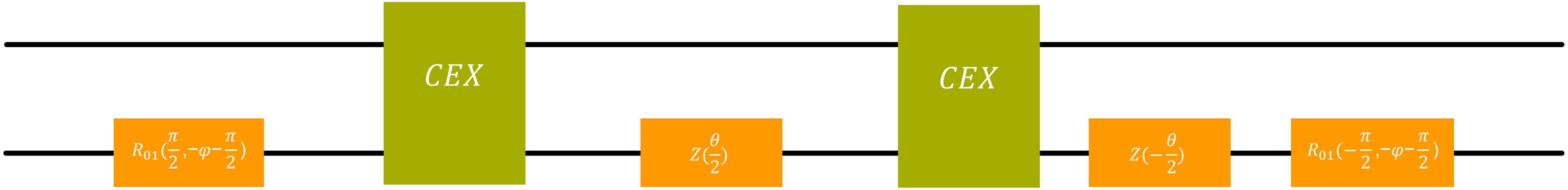


- *Example*:

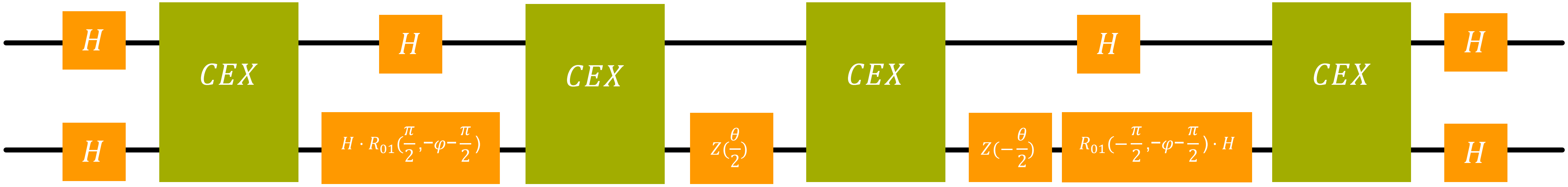
$$\mathrm{cRot}_{2;1,2}(\theta,\phi) = (P_{0,2}\cdot P_{1,2}\otimes P_{0,1}\cdot P_{1,2})^{\dagger}\cdot cRot_{1;0,1}(\theta,\phi)\cdot(P_{0,2}\cdot P_{1,2}\otimes P_{0,1}\cdot P_{1,2})$$

The Basic Brick: CEX Gate

- The decomposition of the chosen ***cRot*** in function of θ and ϕ

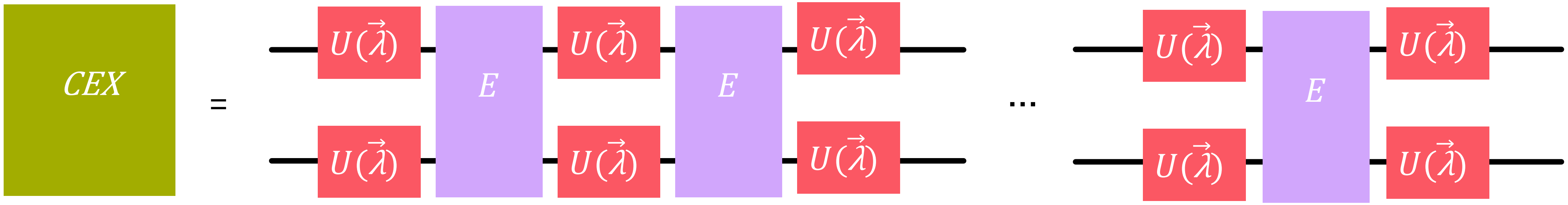


- The decomposition of the chosen ***pSwap*** in function of θ and ϕ



Make your own CEX: Second Step

- Offline
- Parametrized circuit: → *Objective function* → $Fidelity(A, B) = \frac{1}{d^2} Tr\langle A^\dagger , B \rangle$; → *Ansatz Binary Search*



- A genuine qudit entangling gate: *LS gate* $\mathrm{LS}(\theta) = e^{-i\theta\cdot\sum_{i=0}^{d-1} |ii\rangle\langle ii|}$
- Two-level entangling gate: *MS gate* $\mathrm{MS}(\theta) = e^{-i\frac{\theta}{4}\cdot(I\otimes I+\sigma_{x01}\otimes\sigma_{x01})}$
- Custom user input

$$U(\vec{\lambda}) = \left[\prod_{m=0}^{d-2}\left(\prod_{n=m+1}^{d-1} \exp(iZ_{m,n}\lambda_{n,m}) \exp(iY_{m,n}\lambda_{m,n})\right)\right] \cdot \left[\prod_{l=1}^{d-1} \exp(iZ_{l,d}\lambda_{l,l})\right]$$

Expressible representation constructed from $d^2 - 1$ parameters

Case Study

The compilation of $CSUM$:

System	Dim.	$cRot$	$pSwap$	CEX_{tot}	MS_{tot}	LS_{tot}	$(1-F)$
two qutrits	9	44	24	184	1472	368	$< 10^{-4}$
two ququarts	16	92	36	328	9184	10168	$10^{-3} \sim 10^{-4}$

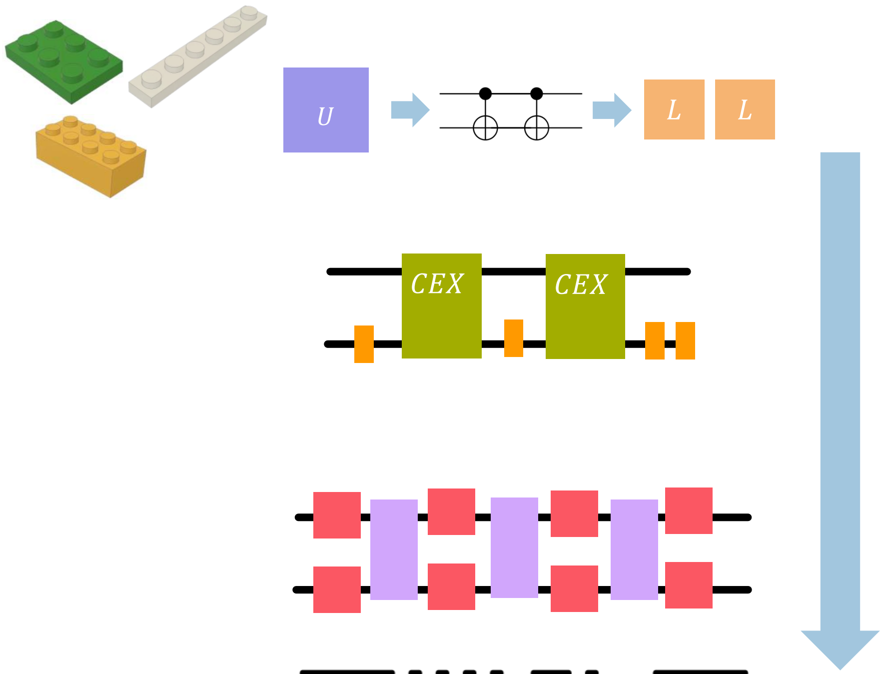
- Feasibility of the fully automated compilation process
- Runtimes in the order of seconds
  for high dimensionality, provided a decomposition for CEX
- Complexity of the solution dominated by the results of QR

Conclusions

- A renewed intuition on qudit entanglement structures
- A decomposition scheme scalable with dimensionality
- Introduced a complete workflow for compiling any two-qudit unitary into any native gate set
- Automated and computationally efficient

Future Work:

- Auxilary qudit levels
- Compression of synthesis results
- Alternative ansatz designs and unitary decompositions
- Optimization routines in post-processing



Tool freely available on Github ☺, as part of the **Munich Quantum Toolkit**

https://github.com/cda-tum/qudit-entanglement-compilation

Questions?

(kevin.mato@tum.de)