

Quantum precomputation: parallelizing cascade circuits and the Moore–Nilsson conjecture is false

Adam Bene Watts¹, Charles R. Chen², J. William Helton², and Joseph Slote^{3,4}

¹University of Calgary

²University of California, San Diego

³University of Washington

⁴California Institute of Technology

Abstract

Parallelization is a major challenge in quantum algorithms due to physical constraints like no-cloning. This is vividly illustrated by the conjecture of Moore and Nilsson from their seminal work on quantum circuit complexity [MN01, announced 1998]: unitaries of a deceptively simple form—controlled-unitary “staircases”—require circuits of minimum depth $\Omega(n)$. If true, this lower bound would represent a major break from classical parallelism and prove a quantum-native analogue of the famous $\text{NC} \neq \text{P}$ conjecture.

In this work we settle the Moore–Nilsson conjecture in the negative by compressing all circuits in the class to depth $\mathcal{O}(\log n)$, which is the best possible. The parallelizations are exact, ancilla-free, and can be computed in $\text{poly}(n)$ time. We also consider circuits restricted to 2D connectivity, for which we derive compressions of optimal depth $\mathcal{O}(\sqrt{n})$.

More generally, we make progress on the project of quantum parallelization by introducing a quantum blockwise precomputation technique somewhat analogous to the method of Arlazarov, Dinič, Kronrod, and Faradžev [Arl+70] in classical dynamic programming, often called the “Four-Russians method.” We apply this technique to more-general “cascade” circuits as well, obtaining for example polynomial depth reductions for staircases of controlled $\log(n)$ -qubit unitaries.

Contents

1	Introduction	3
1.1	Main results	5
1.2	Proof overviews	8
1.3	Outlook	11
1.4	Comments on numerical stability and uniformity	12
1.5	Other related work	12
2	Quantum precomputation	13
2.1	The quantum precomputation identity	13
2.2	Reducing control cascades to nearly diagonal cascades	15
2.3	Stronger parallelization with ancillae	16
3	The Moore–Nilsson conjecture	20
3.1	Partitioning Moore–Nilsson circuits	21
3.2	Valley circuits	22
3.3	A CS decomposition for valley circuits and a refined precomputation identity	23
3.4	Optimal-depth Moore–Nilsson circuits	28
4	Discussion	32
4.1	Open problems near the Moore–Nilsson conjecture	32
4.2	Towards practical precomputation techniques	34
A	Explicit formulas for the CS decomposition of a valley circuit	39
A.1	Notation for this appendix	39
A.2	The formula for the CS Decomposition of a valley	39

1 Introduction

PARALLELISM IS A FUNDAMENTAL ASPECT OF COMPUTATION. On the one hand, finding ways to parallelize algorithms has obvious benefits for runtime in modern computer architectures. On the other, it is desirable that *some* programs do *not* parallelize: various cryptographic primitives rely on the existence of so-called *inherently sequential functions*—polytime functions which do not admit efficient polylog-depth circuits—see for example [BN00; CRR21; Bon+25].¹

But despite its importance, parallelism remains poorly understood. For example, it is open whether every classical bounded fan-in circuit composed of m gates has an equivalent circuit of depth merely $\text{polylog}(m)$ while keeping the gate count to $\text{poly}(m)$. This is essentially the NC vs. P question, a central problem in complexity theory that has stymied the community since its posing in 1981 [Coo81]. While it is generally expected that there are functions in P requiring $\text{poly}(n)$ -depth NC circuits, lower bounds have been stalled for over 30 years at $(3 - o(1)) \log n$, coming from Håstad’s landmark formula lower bounds [Hås93; Hås98].²

Quantum parallelism: constrained by physics?

Parallelism in quantum computation is even more mysterious. Not only are super-logarithmic lower bounds not known, but even basic approaches to upper bounds—borrowed from the classical literature on parallel algorithms—appear to be frustrated by constraints inherent to quantum physics. To see this, consider the very simple toy example pictured in Figure 1.

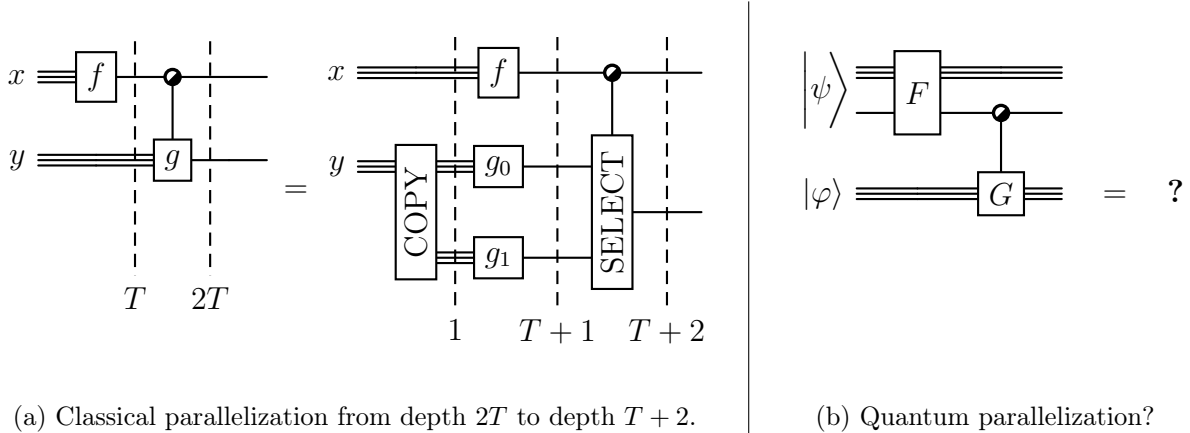


Figure 1: A standard idea in classical parallelization with no immediate quantum analogue. The half-open control denotes the product of an open control and closed control—see Section 1.1 for a formal explanation.

In the classical scenario, we wish to compute a function $h(x, y) := g_{f(x)}(y)$ on two classical inputs x and y for some Boolean functions f, g_0, g_1 . The naive implementation of h is to first

¹We do not attempt a survey of the sizeable literature on time-lock puzzles and related objects initiated by [RSW96]; the cited works provide examples where inherently sequential computations either are provably necessary or at least circumvent previous impossibility results.

²Though there is an important program towards super-logarithmic lower bounds via the KRW conjecture [KRW95]; see the recent work [Mei25] for an overview.

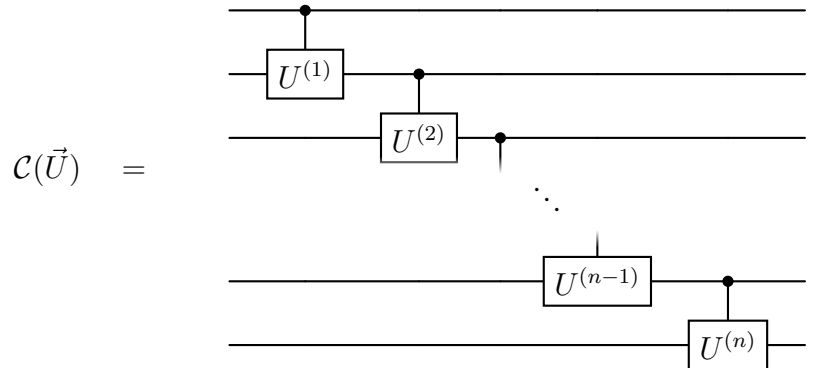
compute $b := f(x) \in \{0, 1\}$, and then run g_b on y . If f , g_0 , and g_1 each take time T to run, this approach takes time about $2T$. But h can be easily parallelized: while one subcircuit is computing $f(x)$, we should copy input y and evaluate $g_0(y)$ and $g_1(y)$ in parallel. Then the final output can be simply selected (in constant time) based on the outcome of f . Here we find parallelization essentially *halves the runtime* to just T plus a small constant overhead. These techniques also extend naturally to reversible classical circuits.³

Now consider a quantum variant: we replace strings x and y with unknown quantum states $|\psi\rangle, |\varphi\rangle$, and our goal is to apply unitary transformations G_0 or G_1 to $|\varphi\rangle$, controlled by the last qubit of $F|\psi\rangle$. Here the parallelization trick no longer works: because of the no-cloning theorem, we cannot copy $|\varphi\rangle$ to simultaneously precompute $G_0|\varphi\rangle$ and $G_1|\varphi\rangle$. We could try to “guess” the outcome of $F|\psi\rangle$, but if we guessed wrongly we would have to rewind our computation on $|\varphi\rangle$ and begin anew. Thus, it seems *this classical parallelization technique has no immediate quantum analogue*.

The Moore–Nilsson conjecture

By iterating the heuristic argument above, one may be led to wonder if there is no way to parallelize a sequence of controlled quantum operations from qubit $j - 1$ to j , $j = 2, \dots, n$. This is precisely the conjecture that concludes the seminal paper of Christopher Moore and Martin Nilsson [MN01], which 25 years ago defined the QNC hierarchy and initiated the study of concrete quantum circuit complexity.

Conjecture (Moore–Nilsson [MN01]). *Let $U^{(1)}, U^{(2)}, \dots, U^{(n)}$ be 1-qubit unitaries which are neither diagonal nor anti-diagonal. Then the following circuit has depth $\Omega(n)$.*



Observe that classical analogues of these circuits can be easily parallelized: for example imagine setting each $U^{(j)}$ to Pauli X . This yields a classical reversible circuit implementing a prefix-sum computation, and such circuits are well-studied and have $\mathcal{O}(\log n)$ depth parallelizations [Ble90; CV89].

The circuit class in the Moore–Nilsson conjecture, however, has resisted parallelization for the quarter century since its first announcement. To the authors’ best knowledge there is no published progress on the conjecture. Through personal communications the authors are aware of some study made of the specific circuit $\mathcal{C}(H, H, \dots, H)$. Chinmay Nirkhe and colleagues

³By adding a layer of uncompute operations to our classical circuit, which can also be performed in parallel after the desired output bit is computed.

derived a circuit that produces $\mathcal{C}(H, H, \dots, H)|0\rangle$ in $\mathcal{O}(\log n)$ depth—though of course this says nothing about the circuit required to implement the entire unitary. Independently, Anne Broadbent considered the circuit $\mathcal{C}(H, H, \dots, H)$ in the context of position verification. This led Florian Speelman and colleagues to note that “middle bits” of the $\mathcal{C}(H, H, \dots, H)$ unitary could be extracted up to small error by log-depth quantum circuits and, as a consequence, that it was possible to construct a log-depth circuit which approximately inverted the action of $\mathcal{C}(H, H, \dots, H)$ on computational basis states. However, it is not clear how to extend this approach to invert the action of $\mathcal{C}(H, H, \dots, H)$ on general states or to obtain an operator-norm approximation of the unitary.

The Moore–Nilsson conjecture highlights dual motivations for studying parallelization in the quantum world:

- The first is the need for *upper bounds*: because Moore–Nilsson circuits are so simple, they underscore how underdeveloped quantum parallelization techniques are in comparison to the classical case. Can we at least recover quantum analogues of basic parallelization ideas? These could lead to new compilation techniques, reducing the runtime of both near-term and fault-tolerant quantum algorithms.
- The second is *lower bounds*: the Moore–Nilsson conjecture presents a simple, concrete class of unitaries that might have super-logarithmic depth lower bounds. Could unitary circuit depth lower bounds join the growing list [Kre21; Kre+23; LMW24, etc.] of quantum hardness results that seem to be available independent of separations (or the lack thereof) in classical complexity?

In this work we develop new techniques for parallelizing quantum circuits. We use these techniques to strongly refute the Moore–Nilsson conjecture, showing Moore–Nilsson circuits can be exactly compiled with depths that match lightcone lower bounds. We also use our parallelization techniques to obtain milder depth reductions for a larger class of circuits consisting of cascades of controlled multi-qubit unitaries. Finally, we discuss new classes of simple unitaries which may still admit super-logarithmic depth lower bounds.

1.1 Main results

We will work with circuits consisting of arbitrary one and two qubit gates with all-to-all connectivity—*i.e.*, QNC circuits⁴—as well as circuits restricted to 2D connectivity, which we call QNC_{2D} circuits. All our results are constructive and can therefore be converted into approximate compilation schemes over a finite gate set using standard techniques.

The first contribution of this paper is a refutation of the Moore–Nilsson conjecture.

Theorem 1.1. *Every Moore–Nilsson unitary $\mathcal{C}(\vec{U})$ on n qubits is computed exactly by...*

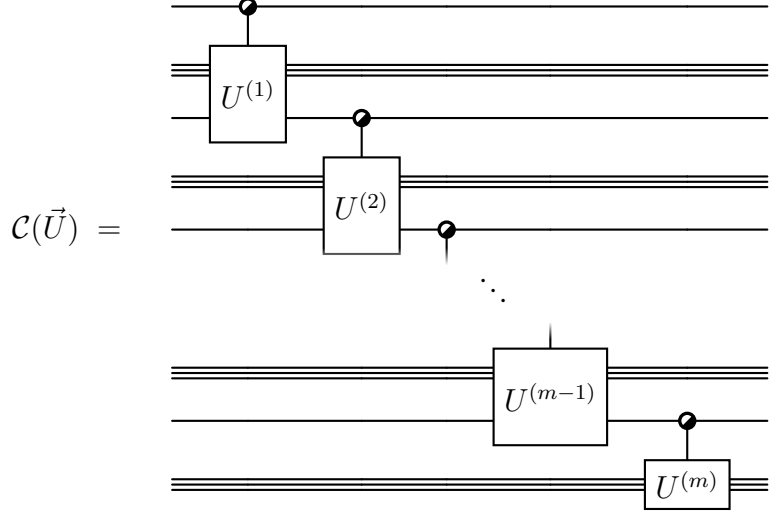
- A QNC circuit of depth $\mathcal{O}(\log n)$ and no ancillae, and
- A QNC_{2D} circuit of depth $\mathcal{O}(\sqrt{n})$ and $\mathcal{O}(n)$ ancilla qubits.

⁴Equivalently (up to constant factors), we consider circuits with a gate set consisting of arbitrary one-qubit gates and at least one entangling two qubit gate [BM03; Bre+02].

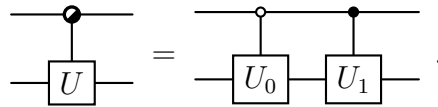
Both of these depths are the best possible. Moreover, these circuits can be computed from the list of gates $U^{(1)}, \dots, U^{(n)}$ in time $\text{poly}(n)$.

Our second result is a milder parallelization theorem that applies to a broader class of circuits which we now define.

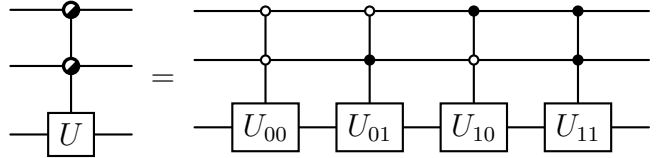
Definition 1.1. Let $\vec{U} = (U_0^{(1)}, U_1^{(1)}, U_0^{(2)}, U_1^{(2)}, \dots, U_0^{(m)}, U_1^{(m)})$ be a list of $2m$ unitaries, each acting on k qubits. Then the control cascade $\mathcal{C}(\vec{U})$ is the $(km + 1)$ -qubit unitary implemented by



Notation: Here and throughout we use the “closed-open” control notation to refer to a product of open and closed controlled unitaries. So, for example:



We extend this notation in the natural way to multiply controlled gates, so



When necessary we will specify the sub-matrices of a closed-open controlled unitary by writing U as a vector of unitaries, *i.e.* $U = (U_0, U_1)$ or $U = (U_{00}, U_{01}, U_{10}, U_{11})$ respectively. In text, we will use the phrase “multiplexer U ” to refer to a unitary of this form.

One may note that, up to a layer of single qubit unitaries, Moore–Nilsson unitaries coincide with the set of control cascades with $k = 1$.⁵ For the larger class of control cascade unitaries we are still able to obtain some depth reductions.

⁵Since we can always rewrite a multiplexer $U = (U_0, U_1)$ as a U_0 unitary followed by a controlled $U_0^\dagger U_1$.

Theorem 1.2. *For any control cascade $\mathcal{C}(\vec{U})$ with m -many k -qubit U 's there is a QNC circuit with depth $\mathcal{O}(4^k + m2^k)$ and no ancillae which exactly computes $\mathcal{C}(\vec{U})$. Moreover, this circuit can be computed from $\mathcal{C}(\vec{U})$ in time $\text{poly}(m, 2^k)$.*

To appreciate this theorem, consider the regime of $m = n$ and $k = \log_2(n)$ (so the unitary $\mathcal{C}(\vec{U})$ acts on $n \log_2(n)$ qubits). Because k -qubit unitaries require up to $\mathcal{O}(4^k)$ gates [Tuc99; Möt+04], and $\mathcal{C}(\vec{U})$ consists of m of these in a cascade, a naive compilation of such unitaries would yield circuits of depth on the order of $m4^k = n^3$. But in the same parameter regime we may apply [Theorem 1.2](#) to get a polynomial improvement to depth $\mathcal{O}(n^2)$.

Corollary 1.3. *Consider $\mathcal{C}(\vec{U})$ with n -many $\log_2(n)$ -qubit gates. Then $\mathcal{C}(\vec{U})$ has an exact, ancilla-free circuit with depth $\mathcal{O}(n^2)$. (C.f. the naive $\mathcal{O}(n^3)$ -depth circuit.)*

We can decrease the depth further by allowing for ancillae and approximations.

Theorem 1.4. *For any $\mathcal{C}(\vec{U})$ with m -many k -qubit unitaries and error threshold ε there exists:*

- (a) *A QNC circuit of depth $\mathcal{O}(4^k + mk)$ and with $m2^k$ ancilla qubits which implements $\mathcal{C}(\vec{U})$ exactly.*
- (b) *A QNC circuit of depth $\mathcal{O}(4^k + m \log \log(m/\varepsilon))$ and with $m \log(m/\varepsilon)$ ancilla qubits which implements a unitary $\mathcal{C}'$ satisfying $\|\mathcal{C}' - \mathcal{C}(\vec{U})\|_\infty \leq \varepsilon$.*

Moreover, these circuits can be computed from $\mathcal{C}(\vec{U})$ in time $\text{poly}(m, 2^k)$.

[Theorem 1.4](#) can give a near-quadratic depth improvement over naive techniques. With $\mathcal{C}(\vec{U})$ composed of n -many $\log_4(n)$ -qubit gates, the naive depth is on the order of $n4^k = n^2$. However direct computation in the same parameter regime gives:

Corollary 1.5. *Consider $\mathcal{C}(\vec{U})$ with n -many $\log_4(n)$ -qubit gates. Then there exists:*

- (a) *A QNC circuit with depth $\mathcal{O}(n \log n)$ and $n^{3/2}$ ancillae implementing $\mathcal{C}(\vec{U})$ exactly.*
- (b) *A QNC circuit with depth $\mathcal{O}(n \log \log n)$ and $\mathcal{O}(n \log n)$ ancillae implementing a unitary $\mathcal{C}'$ satisfying $\|\mathcal{C}' - \mathcal{C}(\vec{U})\|_\infty \leq 2^{-\Omega(n)}$.*

(C.f. the naive $\mathcal{O}(n^2)$ -depth circuit.)

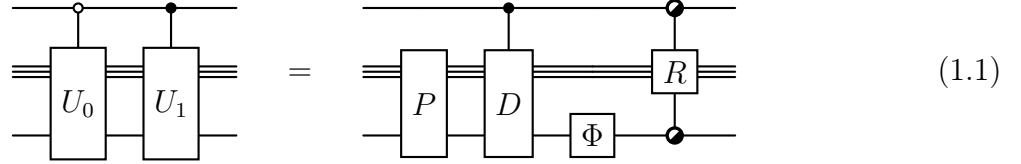
We discuss situations in which our parallelization techniques do not immediately give depth reductions in [Section 1.3](#) of the introduction and then again in [Section 4](#) of the main paper.

1.2 Proof overviews

1.2.1 A quantum precomputation technique

The starting point of our results is to essentially fill in the right-hand side of [Figure 1b](#), though in a weaker way than is achieved classically. Classically, the identity in [Figure 1a](#) splits g into a preprocessing operation independent of the control and then an $\mathcal{O}(1)$ -time controlled operation. In the quantum case, we do not obtain an $\mathcal{O}(1)$ -time controlled operation, but we are able to reduce from a controlled general unitary to a controlled *diagonal* unitary. Concretely, we begin with the following lemma.

Lemma 1.6 (Quantum precomputation identity, in brief). *Let U_0, U_1 be k -qubit unitaries. Then*

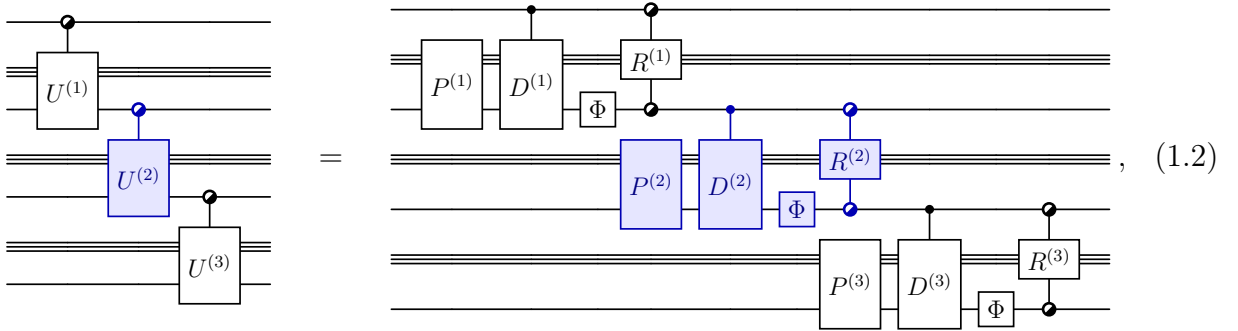


for some unitary P and multiplexer R , diagonal unitary D , and a universal (i.e., fixed) unitary Φ .

The proof of this lemma is given in [Section 2.1](#). It employs a classical result in matrix analysis known as the cosine-sine decomposition (CS decomposition; see *e.g.*, [\[PW94\]](#)), which previously played an important role in quantum circuit compilation [\[Tuc99; Möt+04\]](#) and more recently has provided a valuable perspective on the Quantum Singular Value Transform [\[TT24\]](#).

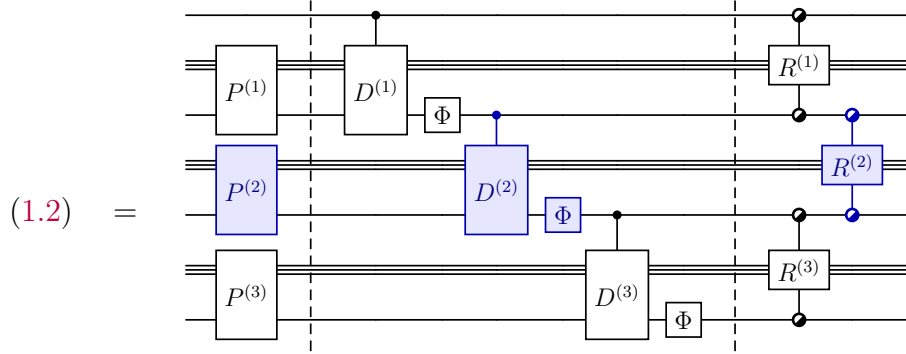
This precomputation identity is the starting point for all our results, which are best explained out of order. [Theorem 1.2](#) follows from applying [Lemma 1.6](#) to each gate in the cascade, as we illustrate now for the case of $m = 3$ controlled unitaries.

To parallelize $\mathcal{C}(\vec{U})$, begin by applying [Lemma 1.6](#) to each controlled unitary $U^{(i)}$'s in the cascade:



where we have highlighted the second unitary and its replacement for visual clarity. The resulting $P^{(i)}$'s act on disjoint qubits so they can be computed in parallel, and the $R^{(i)}$'s only overlap on control qubits and therefore commute. Rearranging based on these observations,

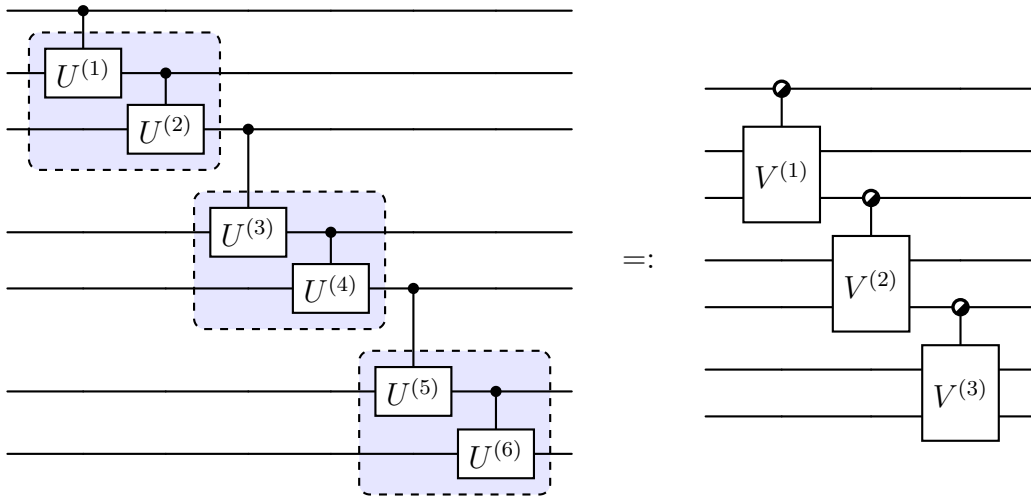
we find the circuit splits into three “stages”:



The $P^{(i)}$'s are k -qubit gates acting in parallel, so by standard circuit compilation results [Tuc99; VMS04] the first stage has a QNC circuit of depth $= \mathcal{O}(4^k)$. The controlled $R^{(i)}$'s can always be organized into two layers, so they also parallelize to depth $\mathcal{O}(4^k)$. Diagonal unitaries on k qubits only ever require $\mathcal{O}(2^k)$ gates [BM04], so the middle section requires depth at most $\mathcal{O}(m2^k)$. Summing these estimates yields [Theorem 1.2](#). A formal proof is given in [Section 2.2](#).

[Theorem 1.4](#) builds on this approach by observing that diagonal unitaries are themselves amenable to classical precomputation techniques. Using ancilla qubits to precompute the “truth tables” of each of the $D^{(j)}$'s in the cascade, either exactly or to finite precision, gives [Theorem 1.4](#). The proof of this theorem appears in [Section 2.3](#).

We remark that theorems [Theorems 1.2](#) and [1.4](#) are already enough to give a weak disproof of the Moore–Nilsson conjecture. The first step is to observe that a controlled cascade of n single-qubit unitaries can also be viewed as a controlled cascade consisting of fewer blocks of multi-qubit unitaries, for example as follows.



Applying [Theorem 1.4](#) to $\mathcal{C}(\vec{V})$ yields a $1/\text{poly}(n)$ operator-norm approximation to the original Moore–Nilsson circuit with depth $\mathcal{O}(n \log \log n / \log n)$ and $\tilde{\mathcal{O}}(n)$ ancillae. This is explained in more detail in [Section 3.1](#).

1.2.2 Aside: comparing with classical work

Before explaining the proof of [Theorem 1.1](#), we pause to mention that the (blockwise) precomputation methods described thus far can be viewed as a quantum analogue of a well-known classical technique for speeding up dynamic programming introduced by the Soviet researchers Arlazarov, Dinič, Kronrod, and Faradžev [[Arl+70](#)]. Arlazarov et al. showed how to speed up exploration of a d -dimensional memo table by precomputing the “truth tables” of small blocks, leading to time complexity improvements from the naive bound of $\mathcal{O}(n^d)$ to the asymptotically better $\mathcal{O}(n^d / \log^{C(d)}(n))$. This approach has come to be known in the algorithms literature as the “Four-Russians Method” or the “Four-Russians speedup” even though only one—Arlazarov—was actually Russian [[Gus97](#)].

That said, the analogy of our quantum results to [[Arl+70](#)] is not yet a complete one. Arlazarov et al. were not considering parallel algorithms (their speedup holds also for standard Turing machines), and when one switches to circuits a few remarks should be made about the difference between depth and gate complexity for computing memo tables. Another point is that the circuits we parallelize—control cascades—are essentially a quantum analogue of *one-dimensional* memo tables. The extent to which the analogy to [[Arl+70](#)] continues into the full setting of many-dimensional memo tables is a tantalizing open question. We expand on these points in the discussion, [Section 4](#).

1.2.3 Optimal-depth circuits for Moore–Nilsson unitaries

It turns out that we can do much better for Moore–Nilsson circuits than described above. The key is to exploit special structure in the precomputation identity that appears when it is applied to 1-qubit control cascades (the $V^{(j)}$ ’s above).

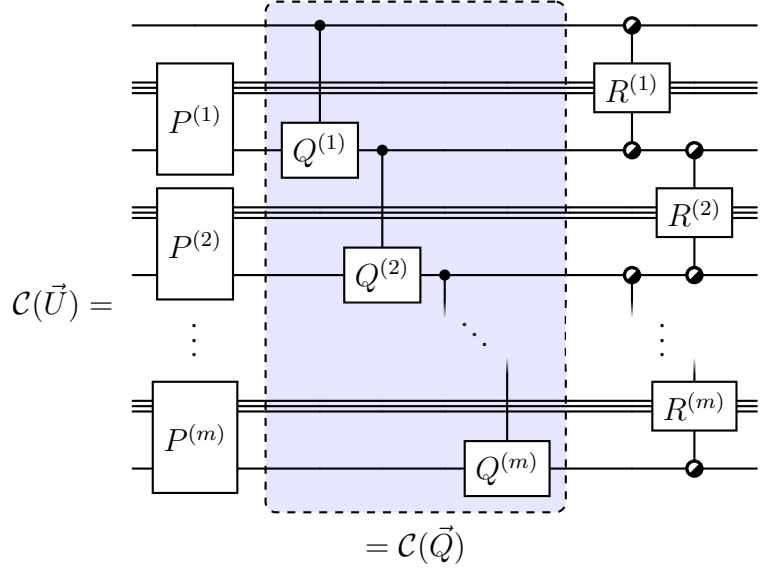
Lemma 1.7 (Quantum precomputation identity for Moore–Nilsson circuits). *Let $U^{(1)}, \dots, U^{(\ell)}$ be any 1-qubit unitaries. Then there exists an ℓ -qubit unitary P , a 1-qubit unitary Q , and a multiplexer R such that*

The diagram shows a quantum circuit with ℓ horizontal qubit lines. On the left, a cascade of unitaries $U^{(1)}, U^{(2)}, \dots, U^{(\ell)}$ is shown. Each $U^{(j)}$ is a 1-qubit gate acting on the j -th qubit, controlled by the j -th qubit of the previous stage. This is equated to a circuit on the right. The right circuit consists of a large block P acting on all ℓ qubits, followed by a single-qubit gate Q on the first qubit, and finally a multiplexer R that routes the output based on the state of the first qubit. The equation is labeled (1.3).

This lemma resembles [Lemma 1.6](#) except that the many-qubit controlled-diagonal gate has been replaced by a single-qubit controlled gate. The proof of [Lemma 1.7](#) is given in [Section 3.3](#), the main technical ingredient of which is a novel cosine-sine-type decomposition specialized to a class of circuits we term “valley circuits.”

To see how [Lemma 1.7](#) leads to [Theorem 1.1](#), let us repeat the procedure above with the stronger precomputation identity. Beginning with a Moore–Nilsson circuit, apply [Lemma 1.7](#) to each of the $V^{(j)}$ ’s and then commute the pre- and postprocessing unitaries to the left and

right of the circuit as in (1.2). We obtain a circuit of the following form:



We see a Moore–Nilsson circuit has appeared in the center column, this time on a small fraction of the original n qubits. Choosing a block size of $\mathcal{O}(1)$ and iterating this procedure $\mathcal{O}(\log n)$ times yields a QNC circuit of total depth $\mathcal{O}(\log n)$, as claimed in [Theorem 1.1](#). To obtain a QNC_{2D} circuit, one carefully arranges the log-depth QNC circuit into a 2D grid, borrowing some tree embedding ideas from VLSI design [[PRS81](#); [RS81](#)]. The formal proof is given in [Section 3.4](#).

1.3 Outlook

The present work offers techniques to reduce the depth of quantum control cascade circuits in various parameter regimes. This is good news for circuit compilation and quantum algorithms. On the other hand, these parallelization results threaten the hope—first articulated by Moore and Nilsson—that it might be easier to resolve the unitary analogue of NC vs. P because even basic quantum circuits seemingly cannot be parallelized.

Our results here indicate that much remains to be understood. While the main results tell us parallelization is sometimes easier than previously thought, our approach does not immediately capture a variety of generalizations of Moore–Nilsson circuits, such as:

- *Cascade circuits with more controls.* Our *control cascade* circuits have one qubit of control. As soon as more qubits—or even one *qutrit*—is used to control the next unitary, our techniques do not seem to immediately apply. For example, in the case of a cascade of qutrit-controlled unitaries, repeating the ideas we used above seems to require a certain 3×3 generalization of the CS decomposition—and as we show in the discussion, [Section 4](#), this generalization is false.
- *Multidimensional “quantum memo tables.”* Control cascade circuits are in some sense a quantum analogue of a one-dimensional memo table. What about two- or higher-dimensional “quantum dynamic programming”? How far does the analogy to Arlazarov et al. [[Arl+70](#)] go?

- *Non-control cascades.* Our techniques also do not seem to immediately apply when control- U is replaced by a general $(k + 1)$ -qubit unitary rather than k -qubit *controlled* unitaries.

These items are discussed in more detail in [Section 4](#). The authors are hopeful that studying these generalizations will lead either to further developments in quantum parallelization methods or, potentially, to super-logarithmic unitary depth lower bounds.

1.4 Comments on numerical stability and uniformity

Our algorithms are exact and polynomial time in the model of real valued computation (BSS) with access to an SVD oracle (or more accurately, a CS decomposition oracle). There are efficient and backwards-stable algorithms for computing the CS decomposition [\[GNS18\]](#). A full discussion of numerical particulars is beyond the scope of this work, but from this we may conclude the existence of an “essentially exact” polytime algorithm for standard Turing machines working to finite precision via the above paper on CS decompositions. Here “essentially exact” means the runtime is polynomial in both the number of qubits n and $\text{polylog}(1/\eta)$ where η is the desired output precision in operator norm.

1.5 Other related work

There is a preexisting work on “Quantum Dynamic Programming” [\[Son+25\]](#), though their notion is different. In that work the next unitary in a sequence is not controlled by the previous state in the sense of being block diagonal in tensor product space, but instead is an arbitrary function of the previous state. In this setting there is a great deal of difficulty in getting efficient circuits for even the naive staircase-type implementation. The authors of [\[Son+25\]](#) show circuits of this form can be implemented approximately in linear depth, provided they have access to an exponential number of ancillae.

Another recent result [\[Kah+25\]](#) shows parallelizations in which “staircase-like” circuits make an appearance, this time for the Quantum Fourier Transform (QFT). The authors of [\[Kah+25\]](#) take advantage of the “staircase-like” form of the approximate QFT to approximately commute subcircuits past each other, deriving an ancilla-free, logarithmic-depth circuit which implements the QFT up to small error in *Frobenius* norm. Then they show a logarithmic-depth worst-to-average case reduction circuit for the QFT, which together with the previous yields a logarithmic-depth operator norm approximation to the QFT with $\mathcal{O}(n/\log n)$ ancillae. It is an interesting open question whether the techniques in [\[Kah+25\]](#) can be adapted to the setting of the current paper or vice versa. Naively, however, the settings appear incompatible: we derive exact or approximate parallelizations in the operator norm directly for our class of circuits, and it is unclear whether shallow worst-to-average case reductions are available for control cascade circuits. In the other direction, our results only apply to staircase-like circuits with one qubit of control, while the staircase encountered in [\[Kah+25\]](#) has multi-qubit controlled phase rotations from one step to the next.

Acknowledgments

Adam Bene Watts and Joseph Slote are very grateful to Atul Singh Arora for introducing the Moore–Nilsson conjecture to us. Arora arrived at the Moore–Nilsson conjecture independently during his work on depth hierarchy theorems relative to an oracle [Aro+23; AGS22]. Adam Bene Watts and Joseph Slote are also very grateful for several discussions about this problem with Henry Yuen and the Columbia quantum group. Adam Bene Watts would additionally like to thank Umesh Vazirani, Richard Cleve, Chinmay Nirkhe, and Peter Høyer for helpful comments about precursors to this work. Parts of this project were completed while Joseph Slote was supported by Chris Umans’ Simons Investigator grant.

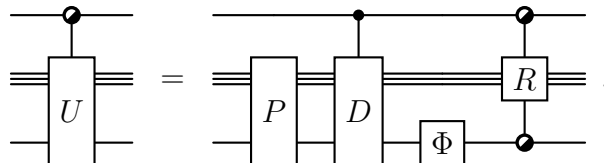
2 Quantum precomputation

This section proves results about parallelizing general control cascade circuits. [Section 2.1](#) proves [Lemma 1.6](#), a quantum precomputation identity that will be a key ingredient in subsequent proofs. [Section 2.2](#) proves [Theorem 1.2](#), which shows we can rewrite a general control cascade as a control cascade of diagonal unitaries sandwiched between pre- and postprocessing operations. [Section 2.3](#) discusses techniques for reducing the depth of the control cascade of diagonal unitaries by introducing ancillae, proving [Theorem 1.4](#).

2.1 The quantum precomputation identity

We begin by stating [Lemma 1.6](#), with more detail than was included in the introduction.

Lemma 1.6 (Repeated). *Let $U = (U_0, U_1)$ be a k -qubit controlled unitary. Then there exists $(k - 1)$ -qubit unitary P , $(k - 1)$ -qubit diagonal unitary D , and multiplexer R unitary on $k + 1$ qubits with two controls such that*



Here $\Phi := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix} = \sqrt{Z}H\sqrt{Z}$.

Moreover, the diagonal matrix D has the following explicit form. Let $W = \begin{pmatrix} W_{00} & W_{01} \\ W_{10} & W_{11} \end{pmatrix}$ be the 2×2 block matrix corresponding to $\text{Rev}(U_0^\dagger U_1)$, i.e., $U_0^\dagger U_1$ with qubit order reversed. Then

$$D = \text{Rev} \begin{pmatrix} \Sigma_1 - i\Sigma_2 & 0 \\ 0 & \Sigma_1 + i\Sigma_2 \end{pmatrix}$$

where Σ_1 is a diagonal matrix of the singular values of W_{00} in ascending order and Σ_2 is a diagonal matrix of the singular values of W_{01} in descending order.

The proof of this lemma relies on a classical SVD-type decomposition for 2×2 block matrices called the *cosine-sine decomposition* (CS decomposition). We state a version of this decomposition for unitaries.

Lemma 2.1 (Cosine-sine decomposition). *Consider any $2N \times 2N$ unitary*

$$U = \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix}.$$

Then there exist $N \times N$ unitaries S_0, S_1, T_0, T_1 such that

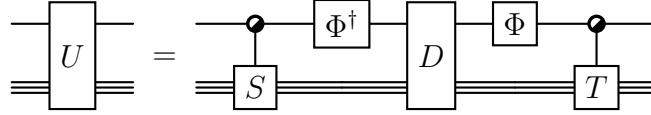
$$U = \begin{pmatrix} S_0 & 0 \\ 0 & S_1 \end{pmatrix} \begin{pmatrix} \Sigma_1 & \Sigma_2 \\ -\Sigma_2 & \Sigma_1 \end{pmatrix} \begin{pmatrix} T_0 & 0 \\ 0 & T_1 \end{pmatrix}$$

where Σ_1 is a diagonal matrix of the singular values of U_{00} in ascending order, and Σ_2 is a diagonal matrix of the singular values of U_{01} in descending order. Moreover, given Σ_2 , this equivalence can be found exactly in time $\text{poly}(N)$.

The CS decomposition has a long history. The name and full statement is essentially due to Chandler Davis and William Kahan [DK69], though ideas along these lines can be traced back as far as the work of Jordan from 1875 [Jor75]. We refer the reader to [PW94] for a historical overview.

The CS decomposition admits a quantum circuit interpretation.

Corollary 2.2 (Cosine-sine circuit decomposition). *Any k -qubit unitary U can be written as*



with $\Phi = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}$, D a diagonal matrix, and multiplexers S and T .

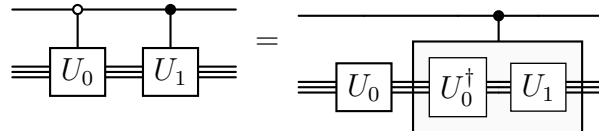
Proof. From Lemma 2.1, we set $\bullet \text{---} [S] = \begin{pmatrix} S_0 & 0 \\ 0 & S_1 \end{pmatrix}$, $\bullet \text{---} [T] = \begin{pmatrix} T_0 & 0 \\ 0 & T_1 \end{pmatrix}$, and $D = \begin{pmatrix} \Sigma_1 - i\Sigma_2 & 0 \\ 0 & \Sigma_1 + i\Sigma_2 \end{pmatrix}$. It is then straightforward to check that

$$(\Phi^\dagger \otimes I_{2^{k-1}}) D (\Phi \otimes I_{2^{k-1}}) = \begin{pmatrix} \Sigma_1 & \Sigma_2 \\ -\Sigma_2 & \Sigma_1 \end{pmatrix},$$

and we are done. □

We now move on to the proof of Lemma 1.6.

Proof (Lemma 1.6): We begin by rewriting $\bullet \text{---} [U]$ as a circuit with a single control gate by “guessing” that U_0 is applied and then undoing if in fact the control wire is 1:



Applying the CS decomposition upside-down to $U_0^\dagger U_1$ gives

$$\begin{array}{c} \circ \\ \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \bullet \\ \text{---} \\ \text{---} \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \bullet \\ \text{---} \\ \text{---} \\ \text{---} \end{array} \quad (2.1)$$

for some S, T and diagonal D . Next the outer control is distributed to the gates inside its target and the $\bullet \text{---} \boxed{S} \text{---} \bullet$ gate is rewritten so that it is open-controlled. It then commutes with the gates to its right, making way for the final rearrangement.

☐

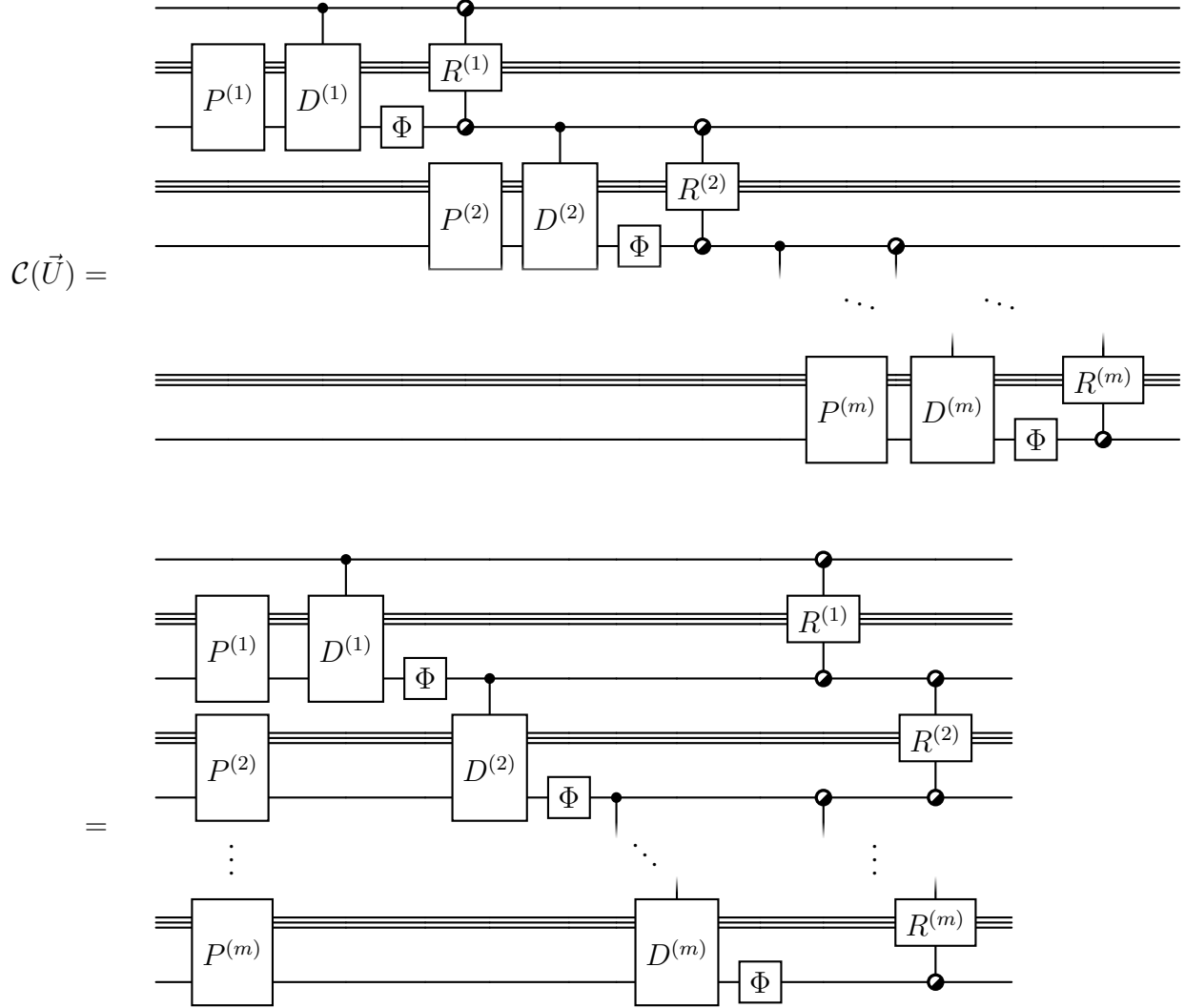
2.2 Reducing control cascades to nearly diagonal cascades

Now we move on to the proof of [Theorem 1.2](#), the key ideas of which were sketched in the introduction.

Theorem 1.2 (Repeated). *For any control cascade $\mathcal{C}(\vec{U})$ with m -many k -qubit U 's there is a QNC circuit with depth $\mathcal{O}(4^k + m2^k)$ and no ancillae which exactly computes $\mathcal{C}(\vec{U})$.*

Proof. Recall that the control cascade $\mathcal{C}(\vec{U})$ consists of a sequence of m closed-open controlled unitaries, each acting on $k + 1$ bits, with the first bit of each closed-open controlled unitary (the control bit) overlapping the last bit of the unitary previous. See [Definition 1.1](#) for a picture. Applying [Lemma 1.6](#) to each controlled unitary in this cascade and then collecting like terms produces three layers of unitaries: a layer of k qubit “preprocessing” unitaries $P^{(i)}$,

a cascade of k -qubit diagonal unitaries and single qubit rotations, and then a layer of doubly controlled $k + 1$ qubit “postprocessing” unitaries $R^{(i)}$:



The $P^{(i)}$ gates each act on disjoint sets of qubits, and the $R^{(i)}$ gates commute and so can be organized into two layers, with each layer involving gates acting on disjoint sets of qubits (as illustrated above). Standard upper bounds show that both these layers can be implemented in depth $\mathcal{O}(4^k)$ [Tuc99; VMS04]. Additionally, any k -qubit diagonal unitary can be implemented exactly by a QNC circuit using at most $\mathcal{O}(2^n)$ gates [BM04], so the central cascade requires depth $\mathcal{O}(m2^k)$. Adding these depths together completes the proof. $\square$

2.3 Stronger parallelization with ancillae

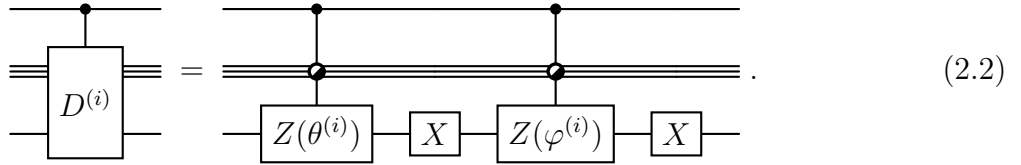
This section focuses on the cascade of diagonal unitaries and single qubit rotations produced by [Theorem 1.2](#). We show this cascade can be further parallelized if ancilla qubits are introduced. Unlike the previous subsections, the techniques used here are standard, though

some care is taken in their application. The result of this parallelization is [Theorem 1.4](#), which we restate now.

Theorem 1.4 (Restated). *For any $\mathcal{C}(\vec{U})$ with m -many k -qubit unitaries and error threshold ε there exists:*

- (a) *A QNC circuit of depth $\mathcal{O}(4^k + mk)$ and with $m2^{k+1}$ ancilla qubits which implements $\mathcal{C}(\vec{U})$ exactly.*
- (b) *A QNC circuit of depth $\mathcal{O}(4^k + m \log(\log(m/\varepsilon)))$ and with $4m \log(m/\varepsilon)$ ancilla qubits which implements a unitary $\mathcal{C}'$ satisfying $\|\mathcal{C}' - \mathcal{C}(\vec{U})\|_\infty \leq \varepsilon$.*

Proof. From the proof of [Theorem 1.2](#), we know that $\mathcal{C}(\vec{U})$ can be written as a layer of pre- and postprocessing unitaries with depth $\mathcal{O}(4^k)$, surrounding a cascade of controlled-diagonal unitaries and single qubit rotations, with each diagonal unitary acting on k qubits. To analyze this cascade, we first note that any diagonal gate $D^{(i)}$ can be written as a product of two multiplexer phase gates acting on a single qubit:



$$D^{(i)} = \text{Circuit with two ancilla qubits and two controlled phase gates } Z(\theta^{(i)}) \text{ and } Z(\varphi^{(i)}) \text{ followed by } X \text{ gates.} \quad (2.2)$$

To formalize this notation, let $\theta^{(i)}, \varphi^{(i)}$ both be vectors of length 2^{k-1} , and let their components $\theta_x^{(i)}, \varphi_x^{(i)}$ be indexed by a length $k-1$ bit strings x . Then the circuit indicates that the product of phase gates

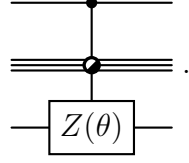
$$\begin{aligned} Z(\theta_x^{(i)})XZ(\varphi_x^{(i)})X &:= \begin{pmatrix} 1 & 0 \\ 0 & \exp(2\pi i \theta_x^{(i)}) \end{pmatrix} X \begin{pmatrix} 1 & 0 \\ 0 & \exp(2\pi i \varphi_x^{(i)}) \end{pmatrix} X \\ &= \begin{pmatrix} \exp(2\pi i \varphi_x^{(i)}) & 0 \\ 0 & \exp(2\pi i \theta_x^{(i)}) \end{pmatrix} \end{aligned}$$

is implemented on the bottom qubit controlled on the $k-1$ qubits above it being in the computational basis state $|x\rangle$ and the upper qubit being in the state $|1\rangle$.⁶

Now the controlled diagonal gates appearing in the central cascade in the proof of [Theorem 1.2](#) can be rewritten as products of multiplexer phase gates using [Equation \(2.2\)](#). This produces a cascade of products of multiplexer phase gates and single qubit rotations. We will show how to approximately and exactly parallelize cascades of this form, proving claims (b) and (a) of [Theorem 1.4](#), respectively. We begin with the approximate parallelization technique.

⁶We use the notation $Z(\theta_x)$ instead of the more standard $P(\theta_x)$ for phase gates to avoid confusing with our preprocessing unitaries $P^{(i)}$.

Proof of Item (b): For notational convenience, we focus on just a single multiplexer phase gate, which we write as



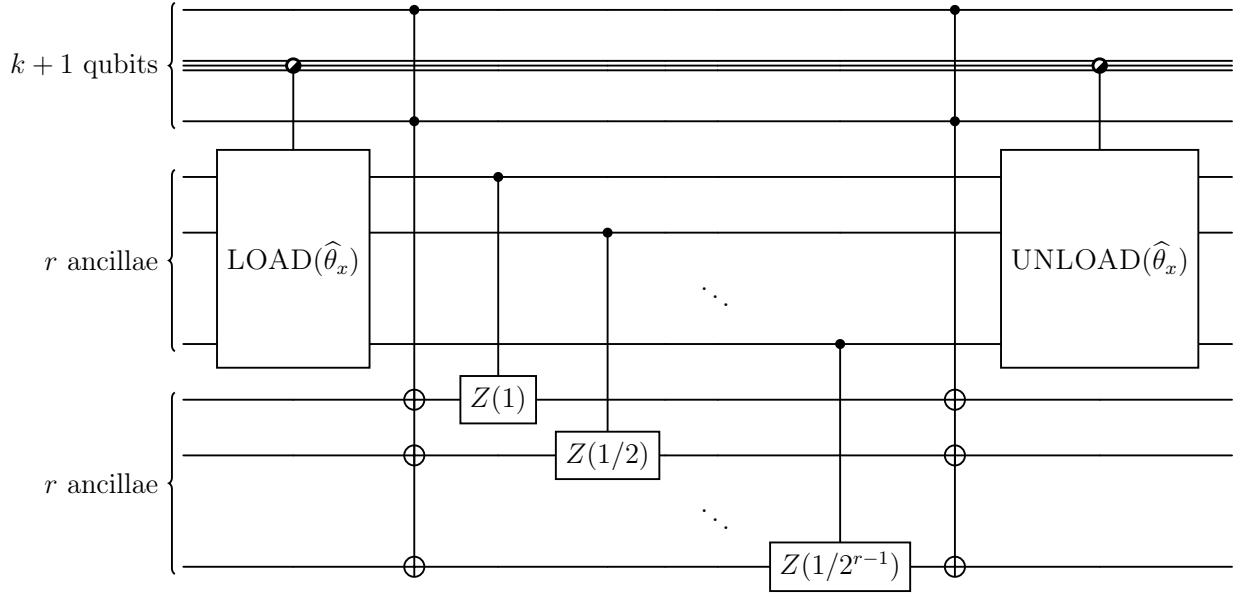
As above, this multiplexer phase gate implements one of 2^{k-1} phase gates $Z(\theta_x)$ on the final qubit, controlled on the computational basis state of the above qubits. Without loss of generality, we can assume that every $\theta_x \in [0, 1)$. Then, for every k bit string x , let $\hat{\theta}_x$ be the approximation to θ_x obtained by taking the first r digits of its binary expansion. We write this as $\theta_x = 0.\theta_{x,1}\theta_{x,2}\dots\theta_{x,r}$.

Next, we define the unitary $\text{LOAD}(\hat{\theta})$ which acts on k control qubits and r ancilla qubits and loads a binary representation of $\hat{\theta}_x$ as

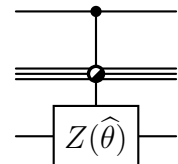
$$\text{LOAD}(\hat{\theta})|x\rangle|0\rangle = |x\rangle|\theta_{x,1}\theta_{x,2}\dots\theta_{x,r}\rangle.$$

We write the inverse operation as $\text{UNLOAD}(\hat{\theta})$. In circuit diagrams, we indicate both LOAD and UNLOAD operations by multiplexer controls indicating the control qubits, and $\text{LOAD}/\text{UNLOAD}$ gates on the target (ancilla) qubits.

Now we consider the following circuit, consisting of $k + 1$ qubits and $2r$ ancillae:



Straightforward calculation of the “phase-kickback” (see, for example, Section 3.1 of [CW00]) shows that this circuit implements a multiplexer $Z(\hat{\theta})$ operation on the first $k + 1$ qubits. That is, the action of the circuit above is equivalent to acting with the circuit

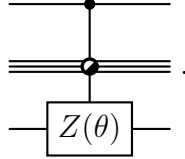


on the first $k + 1$ qubits. But for any x , we also have $\|Z(\theta_x) - Z(\hat{\theta}_x)\|_\infty \leq 2^{-r}$ and so we also have that the LOAD/UNLOAD circuit approximates our original multiplexer phase gate to error at most 2^{-r} in infinity norm.

Now we consider the original cascade of controlled diagonal gates and single qubit rotations. We replace each controlled diagonal gate by multiplexer phase gates using Equation (2.2), then approximate each multiplexer phase gates using $2r$ ancillae and the technique described above. The resulting LOAD operations all commute with each other and can be commuted to the front of the circuit and implemented in depth $\mathcal{O}(2^k)$. Similarly the UNLOAD operations commute with each other and the single qubit rotations (they act on disjoint qubits from the single qubit rotations) and can be commuted to the end of the circuit and also implemented in depth $\mathcal{O}(2^k)$. What is left to implement is the cascade is m iterations of the r -output Toffoli gates and two qubit controlled Z rotations in the above figure. The Toffoli gates can each be implemented with depth $\log r$ using a standard binary-tree fanout, and the controlled Z rotations can be implemented with constant depth.

Then this method lets us approximate the cascade of controlled-diagonal unitaries and single qubit rotations in depth $\mathcal{O}(2^k + m \log r)$. Substituting this into the proof of Theorem 1.2 gives an overall circuit depth of $\mathcal{O}(k^2 4^k + m \log r)$ which implements an approximate version of the original unitary cascade. The approximation error is $\mathcal{O}(m 2^{-r})$ in infinity norm, since there are $2m$ multiplexed phase gates $Z(\theta)$ which were approximated by controlled phase gates $Z(\hat{\theta})$. We also require a total of $4mr$ ancillae to implement all the approximate phase gates. Setting $r = \log(m/\varepsilon)$ completes the proof.

Proof of Item (a): As in the proof of Item (b), we begin by considering a single multiplexer phase gate, which we write as

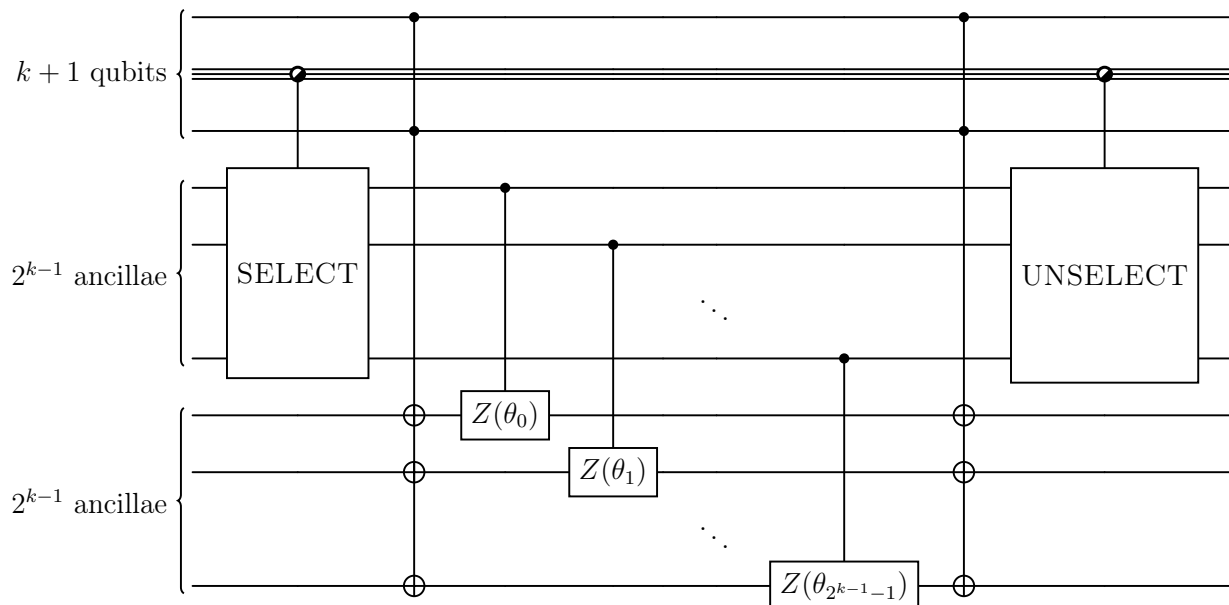


Now we define the SELECT operation, which acts on $k - 1$ qubits and 2^{k-1} ancillae by writing a 1 on the ancilla bit indexed by the computational basis state of the qubits, that is

$$\text{SELECT}|x\rangle|00\dots 0\rangle = |x\rangle|\delta_{0,x}\delta_{1,x}\dots\delta_{2^{k-1}-1,x}\rangle$$

where δ is a Kronecker delta and we interpret the bit string x as indexing an integer between 0 and $2^{k-1} - 1$. We also let UNSELECT be the inverse operation. As with the LOAD/UNLOAD operations, we indicate these operations in a circuit by multiplexer controls on the control qubits, and SELECT/UNSELECT gates on the ancillae.

Now we consider the following circuit:



where we have again identified $k - 1$ bit strings with their integer representations in indexing the phase gates $Z(\theta_x)$. Direct calculation of the phase-kickback shows that this circuit exactly implements the desired multiplexer phase gate on the first $k + 1$ qubits.

Then, as in the approximate case, we can consider the original cascade of controlled diagonal unitaries and single qubit rotations, replace the diagonal gates by multiplexer phase gates, and then rewrite those gates using the technique above. The SELECT and UNSELECT gates can be commuted to the front and back of the circuit, respectively, and implemented in depth $\mathcal{O}(2^k)$ [BM04]. The 2^{k-1} -output Toffoli and phase gates are implemented in a cascade, requiring total depth $\mathcal{O}(mk)$. So the total circuit depth required for this exact rewriting is $\mathcal{O}(4^k + mk)$ while requiring $m2^{k+1}$ total ancillae. $\square$

The reader will notice that the proof above assumes a worst-case $D^{(i)}$. Given some promise about the structure of $D^{(i)}$'s eigenvalues—for example that there is a constant number of them, or that they are generated by a constant number of phases in the circle group—it would be possible to implement the $D^{(i)}$ gates exactly using fewer ancillae and a shallower circuit. This observation is the starting point of the tighter Moore–Nilsson upper bound proven in [Section 3](#).

3 The Moore–Nilsson conjecture

The Moore–Nilsson conjecture concerns circuits $\mathcal{C}(\vec{U})$ where the controlled $U^{(j)}$'s are all single qubit unitaries. This section specializes parallelization techniques from the previous section to this setting and proves tight upper bounds for these Moore–Nilsson circuits. As we will explain, a key ingredient is a CS decomposition for a special class of circuits we term “valley circuits.” It turns out we will only need to use a small amount of the available theory for these circuits; see [Appendix A](#) for a fuller picture of their structure.

3.1 Partitioning Moore–Nilsson circuits

As a first step towards parallelizing Moore–Nilsson circuits, we note that we can collect together cascades of controlled unitaries into groups as pictured in [Figure 2](#). This lets us reinterpret the Moore–Nilsson cascade as a different cascade consisting of fewer multiplexers, each acting on a larger number of qubits.

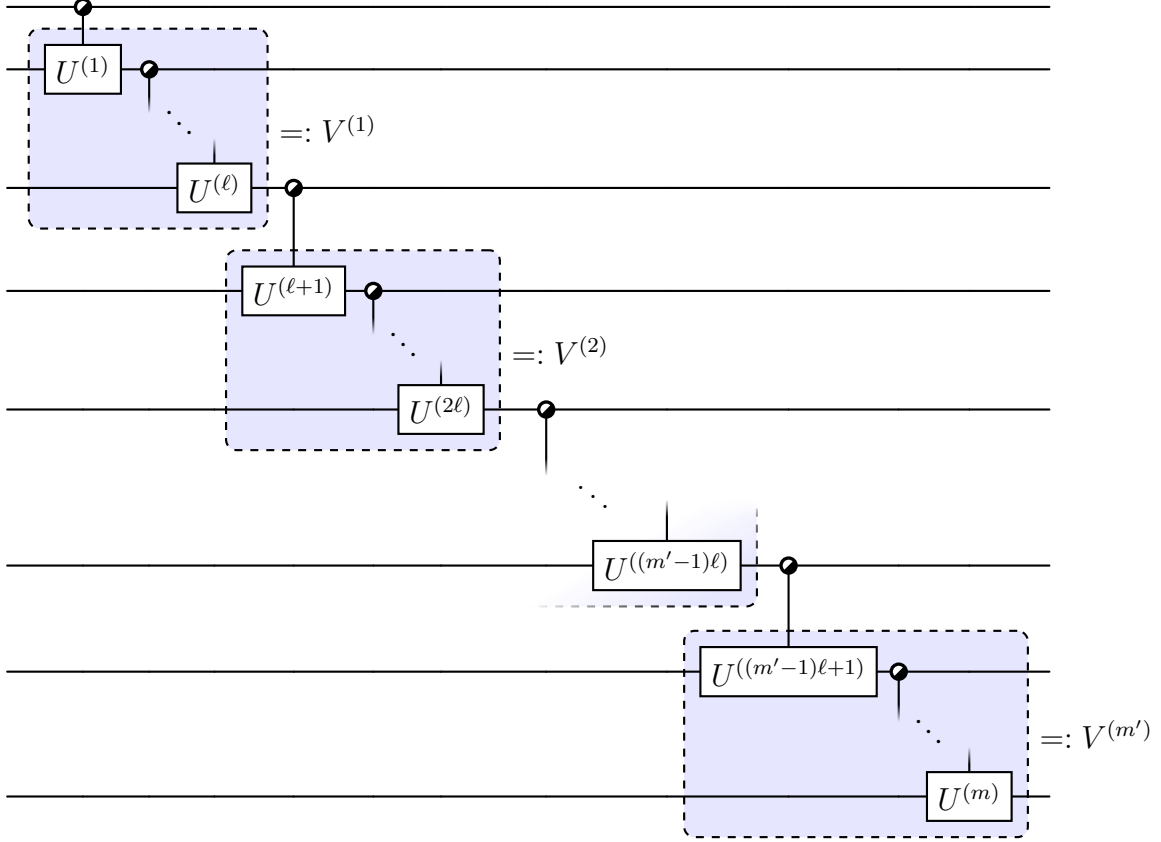


Figure 2: Grouping subcircuits in a Moore–Nilsson circuit to obtain $\mathcal{C}(\vec{V})$, a control cascade circuit of m' -many j -qubit unitaries.

We pause here to remark that for appropriately sized groups, this partitioning of $\mathcal{C}(\vec{U})$ is already sufficient to obtain mild depth reductions for Moore–Nilsson circuits. The key idea is to apply [Theorem 1.4](#) to the each group as follows.

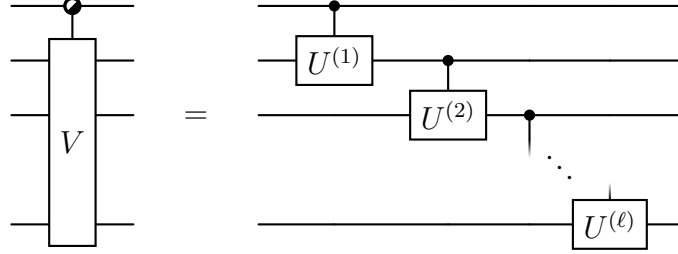
Each unitary in this cascade acts on $k' = \ell k \leq \log(n)/10$ qubits, and there are a total of $m' = m/\ell \leq 100n/\log n$ terms in the cascade. Applying [Theorem 1.4](#) with $\varepsilon = 1/\text{poly}(n)$ gives that this circuit can be implemented in depth

$$\mathcal{O}(4^{k'} + m' \log \log(m'/\varepsilon)) \leq \mathcal{O}\left(n^{0.1} + \frac{100n \log \log(n\varepsilon)}{\log n}\right) = \mathcal{O}\left(\frac{n \log \log n}{\log n}\right).$$

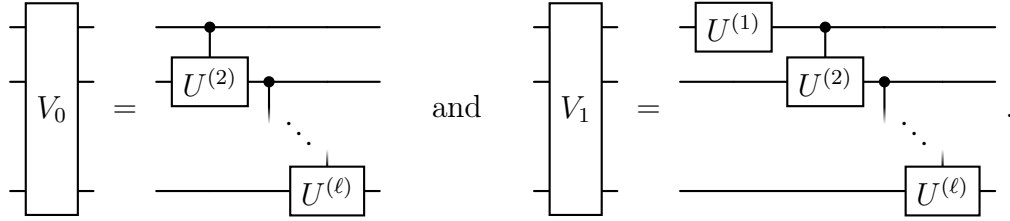
This observation already gives a weak disproof of the Moore–Nilsson conjecture. But we obtain much stronger parallelization results by exploiting the specific structure of the subcircuits formed by this grouping procedure.

3.2 Valley circuits

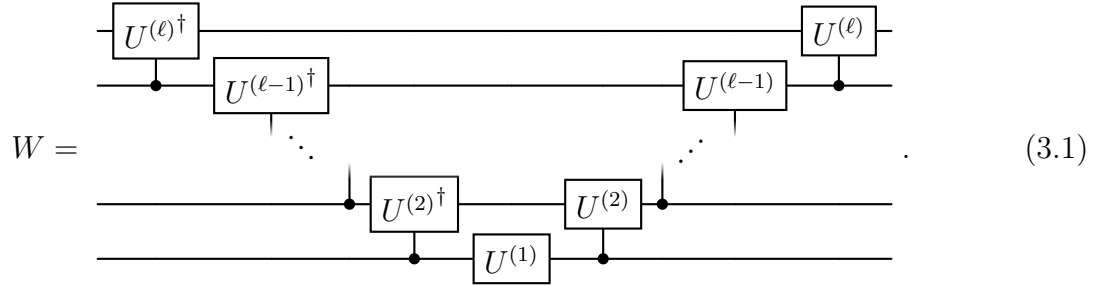
Each of the groups $V^{(j)}$ discussed in the previous section take the form of smaller Moore–Nilsson cascades. In this section we focus on just a single group, which we write as



This multiplexer V is the product of an open-controlled V_0 and closed-controlled V_1 , where V_0 and V_1 are defined as:

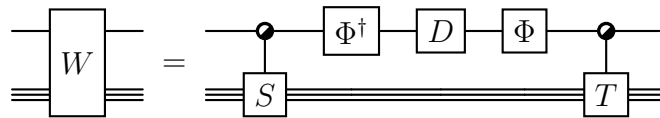


In the previous section, we parallelized cascades by applying the quantum parallelization identity ([Lemma 1.6](#)) to unitaries like the multiplexer V . The proof of this identity involves applying the CS decomposition to the unitary $V_0^\dagger V_1$ with qubit order reversed which we write as $W = \text{Rev}(V_0^\dagger V_1)$. But when V has the form of a Moore–Nilsson cascade, the resulting unitary W also takes on a specific form of a “valley” circuit:



The key technical result underpinning our stronger parallelization results is version of the cosine-sine decomposition adapted specifically to valley circuits. We state this next.

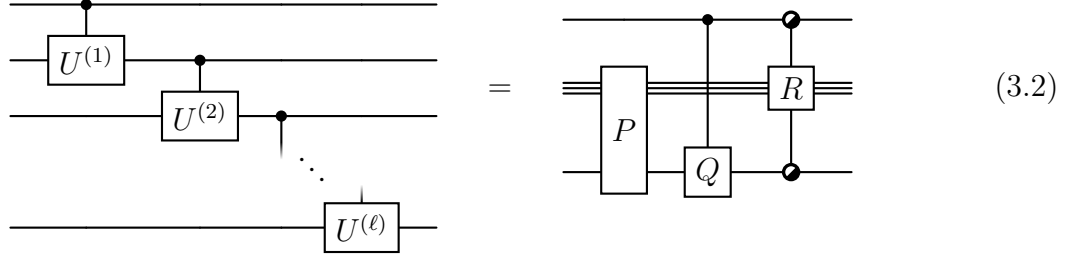
Lemma 3.1 (Cosine-sine for one-qubit valley circuits). *For any circuit W of the form given in [Equation \(3.1\)](#), we also have*



where S, Φ, T are defined as in [Corollary 2.2](#) and D is a single-qubit diagonal unitary.

Using this lemma in place of [Corollary 2.2](#) in the proof of [Lemma 1.6](#) gives the following quantum precomputation identity, first stated in the introduction.

Lemma 1.7 (Repeated). *Let $U^{(1)}, \dots, U^{(\ell)}$ be any 1-qubit unitaries. Then there exists an $(\ell - 1)$ -qubit unitary P , a 1-qubit unitary Q , and a doubly-controlled $(\ell - 2)$ -qubit unitary $R = (R_{00}, R_{01}, R_{10}, R_{11})$ such that*



The next section proves more-general versions of [Lemmas 1.7](#) and [3.1](#), which apply to cascade and valley circuits consisting of multi-qubit controlled unitaries. This generality is not needed for the main result of this section ([Theorem 1.1](#)), but is included because the more-general statements follow naturally from the proof techniques used. A proof of [Theorem 1.1](#) using only [Lemma 1.7](#) is given in [Section 3.4](#) and a reader may skip straight to this section if desired.

3.3 A CS decomposition for valley circuits and a refined precomputation identity

We begin by defining some notation for CS decompositions. For any even-dimensional unitary Y , let a CS decomposition be denoted

$$Y = \begin{pmatrix} S_0^Y & 0 \\ 0 & S_1^Y \end{pmatrix} \begin{pmatrix} \Sigma_1^Y & \Sigma_2^Y \\ -\Sigma_2^Y & \Sigma_1^Y \end{pmatrix} \begin{pmatrix} T_0^Y & 0 \\ 0 & T_1^Y \end{pmatrix},$$

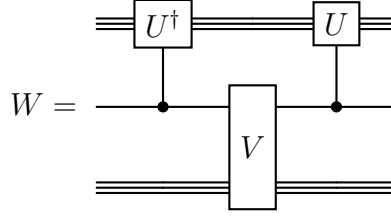
Recall that Σ_1^Y, Σ_2^Y are diagonal matrices. We call these matrices the *stubs* of Y . The stubs of Y are not unique. We can permute the entries of Σ_1^Y, Σ_2^Y simultaneously by conjugating with the permutation

$$I_2 \otimes \pi = \begin{pmatrix} \pi & 0 \\ 0 & \pi \end{pmatrix},$$

which can be absorbed into S and T . Similarly, we can change the phase of the entries of Σ_1^Y, Σ_2^Y be left multiplying by a diagonal matrix, and then absorbing that matrix into the S_0^Y, S_1^Y matrices. Throughout this section we use the convention that the stubs are nonnegative diagonal matrices but entries can be arranged in any order.

The stubs of a valley circuit can be written in terms of the stubs of the unitaries which define the valley circuit. The next lemma gives such an algebraic stub formula. In [Appendix A](#) we expand this result further and give formulas for all of the components S, T and stubs of the CS Decomposition. The appendix proof is also stated in terms of circuit diagrams.

Lemma 3.2. *Let unitaries U , V and W be related as in the following circuit.*



Then, for any CS Decompositions for U , V , and W , defining

$$M := (S_0^U \otimes |0\rangle\langle 0| \otimes T_0^{V\dagger} + T_0^{U\dagger} \otimes |1\rangle\langle 1| \otimes T_1^{V\dagger}) T_0^{W\dagger}.$$

gives

$$(\Sigma_2^W)^2 = M^\dagger ((\Sigma_2^U)^2 \otimes I_2 \otimes (\Sigma_2^V)^2) M.$$

Consequently, there exists a CS Decomposition of W with T_0^W chosen to make $M = I$ and

$$(\Sigma_2^W)^2 = (\Sigma_2^U)^2 \otimes I_2 \otimes (\Sigma_2^V)^2.$$

Proof. To start, we can break W into its blocks:

$$W = \begin{pmatrix} W_{00} & W_{01} \\ W_{10} & W_{11} \end{pmatrix}$$

Next, we want to rewrite W_{10} in terms of U and V . First, using $|0\rangle$ and $|1\rangle$ we can use the middle wire to split the blocks to rewrite W :

$$W = [(U^\dagger \otimes |1\rangle\langle 1| + I_{\dim(U)} \otimes |0\rangle\langle 0|) \otimes I_{\dim(V)/2}] (I_{\dim(U)} \otimes V) [(U \otimes |1\rangle\langle 1| + I_{\dim(U)} \otimes |0\rangle\langle 0|) \otimes I_{\dim(V)/2}].$$

Breaking V into its blocks

$$V = \begin{pmatrix} V_{00} & V_{01} \\ V_{10} & V_{11} \end{pmatrix},$$

we can then rewrite

$$V = |0\rangle\langle 0| \otimes V_{00} + |1\rangle\langle 0| \otimes V_{10} + |0\rangle\langle 1| \otimes V_{01} + |1\rangle\langle 1| \otimes V_{11}.$$

With this, we can multiply out W to get

$$W = I_{\dim(U)} \otimes |0\rangle\langle 0| \otimes V_{00} + U^\dagger \otimes |1\rangle\langle 0| \otimes V_{10} + U \otimes |0\rangle\langle 1| \otimes V_{01} + I_{\dim(U)} \otimes |1\rangle\langle 1| \otimes V_{11}.$$

From here, we can express W_{10} using W :

$$\begin{aligned} W_{10} &= (\langle 1| \otimes I_{\dim(W)/2}) W (|0\rangle \otimes I_{\dim(W)/2}) \\ &= (\langle 1| \otimes I_{\dim(U)/2} \otimes I_2 \otimes I_{\dim(V)/2}) W (|0\rangle \otimes I_{\dim(U)} \otimes I_2 \otimes I_{\dim(V)/2}) \\ &= [(\langle 1| \otimes I_{\dim(U)/2}) U^\dagger (|0\rangle \otimes I_{\dim(U)/2})] \otimes V_{10} + [(\langle 1| \otimes I_{\dim(U)}) U (|0\rangle \otimes I_{\dim(U)})] \otimes V_{01}. \end{aligned}$$

To continue we expand U similarly to W and V as

$$U = \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix}.$$

With this,

$$W_{10} = U_{01}^\dagger \otimes |1\rangle\langle 0| \otimes V_{10} + U_{10} \otimes |0\rangle\langle 1| \otimes V_{01},$$

so

$$W_{10}^\dagger W_{10} = U_{01} U_{01}^\dagger \otimes |0\rangle\langle 0| \otimes V_{10}^\dagger V_{10} + U_{10}^\dagger U_{10} \otimes |1\rangle\langle 1| \otimes V_{01}^\dagger V_{01}.$$

Now we write a CS decomposition of U as

$$U = \begin{pmatrix} S_0^U & 0 \\ 0 & S_1^U \end{pmatrix} \begin{pmatrix} \Sigma_1^U & \Sigma_2^U \\ -\Sigma_2^U & \Sigma_1^U \end{pmatrix} \begin{pmatrix} T_0^U & 0 \\ 0 & T_1^U \end{pmatrix}$$

so $U_{10} = -S_1^U \Sigma_2^U T_0^U$ and $U_{01} = S_0^U \Sigma_2^U T_1^U$. Likewise, we can write a CS Decomposition of V and W so $V_{10} = -S_1^V \Sigma_2^V T_0^V$, $V_{01} = S_0^V \Sigma_2^V T_1^V$, and $W_{10} = -S_1^W \Sigma_2^W T_0^W$. With this,

$$W_{10}^\dagger W_{10} = S_0^U \Sigma_2^U \Sigma_2^{U\dagger} S_0^{U\dagger} \otimes |0\rangle\langle 0| \otimes T_0^{V\dagger} \Sigma_2^{V\dagger} \Sigma_2^V T_0^V + T_0^{U\dagger} \Sigma_2^{U\dagger} \Sigma_2^U T_0^U \otimes |1\rangle\langle 1| \otimes T_1^{V\dagger} \Sigma_2^{V\dagger} \Sigma_2^V T_1^V.$$

Conjugating $W_{10}^\dagger W_{10}$ by the unitary matrix $S_0^U \otimes |0\rangle\langle 0| \otimes T_0^{V\dagger} + T_0^{U\dagger} \otimes |1\rangle\langle 1| \otimes T_1^{V\dagger}$ we find

$$\begin{aligned} & \left(S_0^U \otimes |0\rangle\langle 0| \otimes T_0^{V\dagger} + T_0^{U\dagger} \otimes |1\rangle\langle 1| \otimes T_1^{V\dagger} \right)^\dagger W_{10}^\dagger W_{10} \left(S_0^U \otimes |0\rangle\langle 0| \otimes T_0^{V\dagger} + T_0^{U\dagger} \otimes |1\rangle\langle 1| \otimes T_1^{V\dagger} \right) \\ &= \Sigma_2^{U\dagger} \Sigma_2^U \otimes |0\rangle\langle 0| \otimes \Sigma_2^{V\dagger} \Sigma_2^V + \Sigma_2^U \Sigma_2^{U\dagger} \otimes |1\rangle\langle 1| \otimes \Sigma_2^{V\dagger} \Sigma_2^V \\ &= \Sigma_2^{U\dagger} \Sigma_2^U \otimes I_2 \otimes \Sigma_2^{V\dagger} \Sigma_2^V \end{aligned}$$

At the same time, $W_{10}^\dagger W_{10} = T_0^{W\dagger} \Sigma_2^{W\dagger} \Sigma_2^W T_0^W$ from the CS Decomposition for W .

Then we set

$$M := \left(S_0^U \otimes |0\rangle\langle 0| \otimes T_0^{V\dagger} + T_0^{U\dagger} \otimes |1\rangle\langle 1| \otimes T_1^{V\dagger} \right) T_0^{W\dagger}$$

so

$$\Sigma_2^{W\dagger} \Sigma_2^W = M^\dagger \left(\Sigma_2^{U\dagger} \Sigma_2^U \otimes I_2 \otimes \Sigma_2^{V\dagger} \Sigma_2^V \right) M.$$

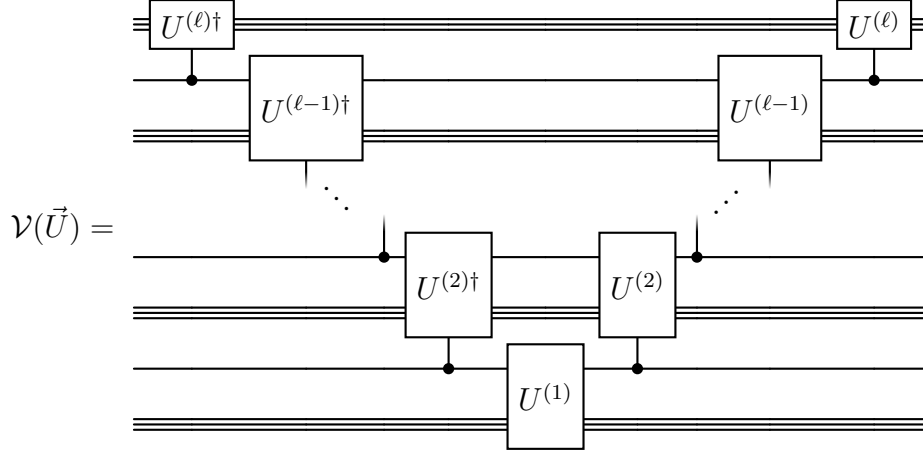
Note to this point we have not used that the Σ_2 are real diagonal matrices. But since they come from the CS Decomposition and are therefore real we have

$$(\Sigma_2^W)^2 = M^\dagger \left((\Sigma_2^U)^2 \otimes I_2 \otimes (\Sigma_2^V)^2 \right) M,$$

as desired. □

Applying the above lemma inductively gives us our desired CS decomposition for valleys. The resulting lemma applies to a larger class of valley circuits in which each of the unitaries $U^{(i)}$ can possibly acting on many qubits, although the single-qubit case remains the strongest version of the statement. We define this class of circuits first, then state the lemma and a useful corollary.

Definition 3.1. Let $\vec{U} = (U^{(1)}, U^{(2)}, \dots, U^{(\ell)})$ be a vector of unitaries, with the number of qubits each unitary acts on not necessarily equal. Then the valley circuit $\mathcal{V}(\vec{U})$ is the unitary given by



Lemma 3.3. For any valley $\mathcal{V}(U^{(1)}, U^{(2)}, \dots, U^{(\ell)})$, and arbitrary CS decompositions of $U^{(1)}, U^{(2)}, \dots, U^{(\ell)}$, there exists a CS decomposition of $\mathcal{V}$ with

$$(\Sigma_2^{\mathcal{V}})^2 = (\Sigma_2^{U^{(\ell)}})^2 \otimes \bigotimes_{j=1}^{\ell-1} [(\Sigma_2^{U^{(\ell-j)}})^2 \otimes I_2].$$

Proof. The proof is by induction on ℓ . For $\ell = 1$, the identity is trivial. Let $\mathcal{V}' = \mathcal{V}(U^{(1)}, U^{(2)}, \dots, U^{(\ell-1)})$ be a valley with the first and last controlled $U^{(\ell)}$ removed. Then, by [Lemma 3.2](#), for any CS decompositions of $\mathcal{V}'$ and $U^{(\ell)}$ there exists a CS decomposition of $\mathcal{F}$ with

$$(\Sigma_2^{\mathcal{V}})^2 = (\Sigma_2^{U^{(\ell)}})^2 \otimes I_2 \otimes (\Sigma_2^{\mathcal{V}'})^2.$$

And, by our induction hypothesis we have that there exists a CS decomposition of $\mathcal{V}'$ with

$$(\Sigma_2^{\mathcal{V}'})^2 = (\Sigma_2^{U^{(\ell-1)}})^2 \otimes \bigotimes_{j=2}^{\ell-1} \left((\Sigma_2^{U^{(\ell-j)}})^2 \otimes I_2 \right)$$

Inserting this expression into the one above completes the proof. $\square$

Corollary 3.4. For a valley $\mathcal{V}(U^{(1)}, U^{(2)}, \dots, U^{(\ell)})$:

1. If $U^{(1)}, U^{(2)}, \dots, U^{(\ell)}$ are all one qubit unitaries, there exists a CS decomposition of $\mathcal{V}$ where $\Sigma_2^{\mathcal{V}}$ is a scalar multiple of the identity.
2. For any $U^{(1)}, U^{(2)}, \dots, U^{(\ell)}$ there exists CS Decompositions of $\mathcal{V}$ with

$$\Sigma_2^{\mathcal{V}} = I_{2^{\ell-1}} \otimes \bigotimes_{j=1}^{\ell} \Sigma_2^{U^{(j)}} \quad \text{or with} \quad \Sigma_2^{\mathcal{V}} = \left(\bigotimes_{j=1}^{\ell} \Sigma_2^{U^{(j)}} \right) \otimes I_{2^{\ell-1}}.$$

Proof. **Item 2** follows immediately from **Lemma 3.3** and non-uniqueness of CS Decompositions under block permutations of the Σ 's. **Item 1** follows from **Item 2** with all $\Sigma_2^{U^{(j)}}$ being 1×1 matrices and therefore scalars. $\square$

We can now prove a refined version of our quantum precomputation identity (**Lemma 1.6**), which bounds the size of the diagonal matrix D in the circuit. As in the case of **Corollary 3.4**, this lemma is stated for cascades with arbitrary-sized controlled unitaries, but the strongest versions of the statement apply to one-qubit cascades.

Lemma 3.5 (Quantum precomputation identity with stub count). *Let $\mathcal{C}(U_0^{(1)}, U_1^{(1)}, \dots, U_0^{(\ell)}, U_1^{(\ell)})$ be a control cascade which acts on a total of d qubits, with unitaries $U_0^{(1)}, U_1^{(1)}, \dots, U_0^{(\ell)}, U_1^{(\ell)}$ arbitrary. Then there exists an $(d - 1)$ -qubit unitary P' , a diagonal $(d - \ell)$ -qubit unitary D' , and a multiplexer R such that*

$$\mathcal{C}(\vec{U}) = \begin{array}{c} \text{Diagram of Lemma 3.5 circuit} \end{array} \quad (3.3)$$

Formulas for all these unitaries are given explicitly given in the proof.

Proof. This result is a refinement of the quantum precomputation identity

$$\mathcal{C}(U_0^{(1)}, U_1^{(1)}, \dots, U_0^{(\ell)}, U_1^{(\ell)}) = \begin{array}{c} \text{Diagram of quantum precomputation identity} \end{array}$$

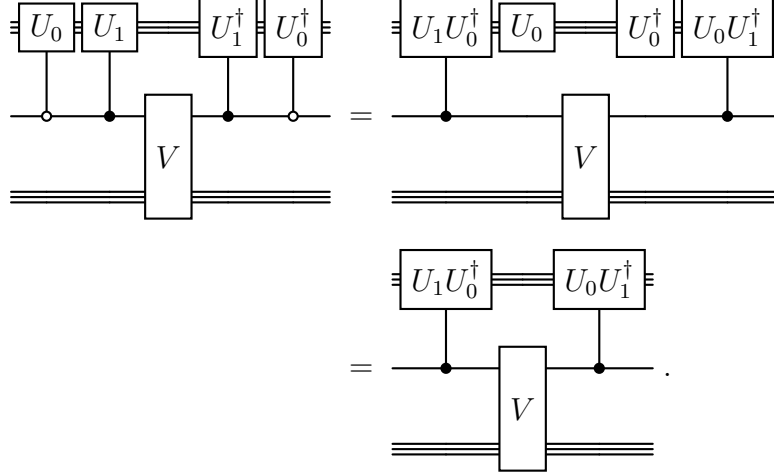
We will show the diagonal matrix D in the quantum precomputation identity factors as

$$D = I_2^{\otimes \ell-1} \otimes D'$$

for a $(d - \ell)$ -qubit diagonal unitary D' .

The D in the quantum precomputation identity is simply the D coming from the CS decomposition of valley with multiplexers $U^{(\ell)}, \dots, U^{(1)}$. But this valley of multiplexers is equivalent to a valley of singly controlled unitaries with the same dimension, since we can

rewrite a multiplexer valley via repeated applications of the identity



Then by [Lemma 3.2](#), we can select a cosine-sine decomposition with the stub

$$\Sigma_2 = I_2^{\otimes \ell-1} \otimes \bigotimes_{j=1}^{\ell} \Sigma_2^{U(j)}.$$

We write this as

$$\Sigma_2 = I_2^{\otimes \ell-1} \otimes \widehat{\Sigma}_2$$

where $\widehat{\Sigma}_2 = \bigotimes_{j=1}^{\ell} \Sigma_2^{U(j)}$. For the same cosine-sine decomposition we have Σ_1 by

$$\Sigma_1 = \sqrt{I_2^{\otimes d-2} - \Sigma_2^2} = \sqrt{I_2^{\otimes d-2} - I_2^{\otimes \ell-1} \otimes \widehat{\Sigma}_2^2} = I_2^{\otimes \ell-1} \otimes \widehat{\Sigma}_1$$

where

$$\widehat{\Sigma}_1 = \sqrt{I_2^{\otimes d-\ell-1} - \widehat{\Sigma}_2^2}.$$

From $D = \begin{pmatrix} \Sigma_1 - i\Sigma_2 & 0 \\ 0 & \Sigma_1 + i\Sigma_2 \end{pmatrix}$, we can factor $I_2^{\otimes \ell-1}$ to get

$$D = \begin{pmatrix} I_2^{\otimes \ell-1} \otimes (\widehat{\Sigma}_1 - i\widehat{\Sigma}_2) & 0 \\ 0 & I_2^{\otimes \ell-1} \otimes (\widehat{\Sigma}_1 + i\widehat{\Sigma}_2) \end{pmatrix} = I_2^{\otimes \ell-1} \otimes D'$$

where

$$D' := \begin{pmatrix} \widehat{\Sigma}_1 - i\widehat{\Sigma}_2 & 0 \\ 0 & \widehat{\Sigma}_1 + i\widehat{\Sigma}_2 \end{pmatrix}.$$

□

3.4 Optimal-depth Moore–Nilsson circuits

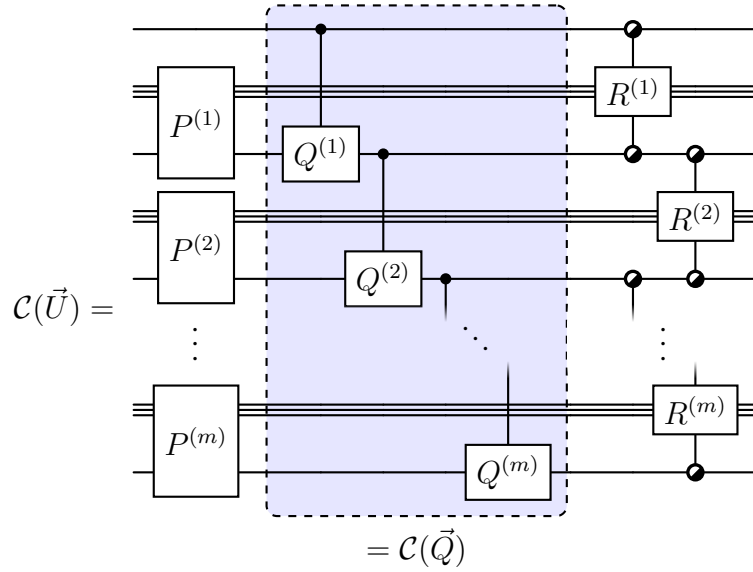
We are now ready to prove our optimal depth compression result for Moore–Nilsson circuits ([Theorem 1.1](#)). We begin by repeating the theorem here, then give its proof.

Theorem 1.1 (Repeated). *Every Moore–Nilsson unitary $\mathcal{C}(\vec{U})$ on n qubits is computed exactly by...*

- A QNC circuit of depth $\mathcal{O}(\log n)$ and no ancillae, and
- A QNC_{2D} circuit of depth $\mathcal{O}(\sqrt{n})$ and $\mathcal{O}(n)$ ancillae.

Both of these depths are the best possible. Moreover, these circuits can be computed from the list of gates $U^{(1)}, \dots, U^{(n)}$ in time $\text{poly}(n)$.

Proof. Group the cascade into blocks $V^{(j)}$ of size $\ell = b$ as pictured in [Figure 2](#), apply the quantum precomputation identity for Moore–Nilsson circuits ([Lemma 1.7](#)), and rearrange to obtain:



The depth of the first column is at most $\mathcal{O}(4^b)$ [[Tuc99](#); [VMS04](#)]; same for the last column. The middle column is a Moore–Nilsson circuit on $\mathcal{O}(n/b)$ qubits. Iterating this identity on successive $\mathcal{C}(\vec{Q})$'s r -many times yields a circuit of total depth at most

$$\mathcal{O}\left(r4^b + \frac{n}{b^r}\right).$$

Setting b to a sufficiently large constant and $r = \log(n)$ we find the depth is at most $\mathcal{O}(\log n)$ for all-to-all connected QNC circuits. This completes the first assertion of [Theorem 1.1](#).

We turn to the second assertion of the theorem and describe how to compile the circuit above into a 2D connectivity architecture with optimal depth. We will work with the concrete choice of block size $b = 2$.

Given any quantum circuit, we call its *forward topology* the directed graph obtained by, for each qubit j , truncating the j^{th} wire so that it terminates at the final gate that interacts with qubit j . Vertices in the resulting graph correspond to gates in the circuit, while edges in the graph correspond to wires and are oriented with the flow of time. Parallel edges (i.e. edges with the same out and in vertex) are replaced with a single edge and a label denoting

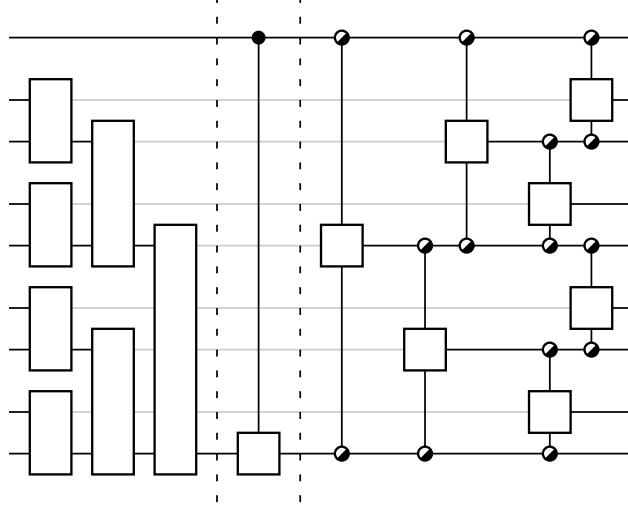


Figure 3: An example 9-qubit Moore–Nilsson circuit after parallelization. It naturally separates into three stages, with the first stage having the forward topology of a complete binary tree.

the number of qubits represented by the edge. The *backward topology* is defined analogously, with time reversed. This is illustrated in the example of a 9-qubit Moore–Nilsson circuit in [Figure 3](#) and [Figure 4](#). Now we argue that the first and last “stages” of the circuit have nice forwards and backwards topologies, respectively.

Claim 1. The P stage of the Moore–Nilsson parallelization has forward topology equal to the complete binary tree. The R stage of the Moore–Nilsson parallelization can be rewritten into a circuit with backward topology equal to the complete binary tree, followed by a layer of CNOTs.

Proof of claim. The first part is immediate. The second part can be seen by splitting shared control wires into left and right sides with CNOTs that can be easily uncomputed in depth one afterwards. See [Figure 4](#). $\diamond$

Claim 2. Fix an embedding of the n -leaf complete binary tree in an m -vertex 2D grid and let d be the maximum edge length in the embedding. Then any quantum circuit with complete binary tree forward topology and where each gate is size $\mathcal{O}(1)$ and each edge is on $\mathcal{O}(1)$ qubits has a compilation into the 2D architecture on $\mathcal{O}(m)$ qubits with depth $\mathcal{O}(d \log n)$.

Proof of claim. We describe a procedure for deriving a 2D quantum circuit from the tree embedding.

We begin by identifying our m -vertex grid with a course-grained version of the 2D architecture of qubits, so that each vertex of the grid corresponds to a constant-sized square of qubits. We set this constant at least as large as the maximum number of qubits involved in a single gate in our original circuit.

Now the gates corresponding to each level of the tree will be computed in parallel as a different stage (group of consecutive layers) of the quantum circuit. Each gate is implemented on the subset of qubits identified by the embedding. Between the gates corresponding to

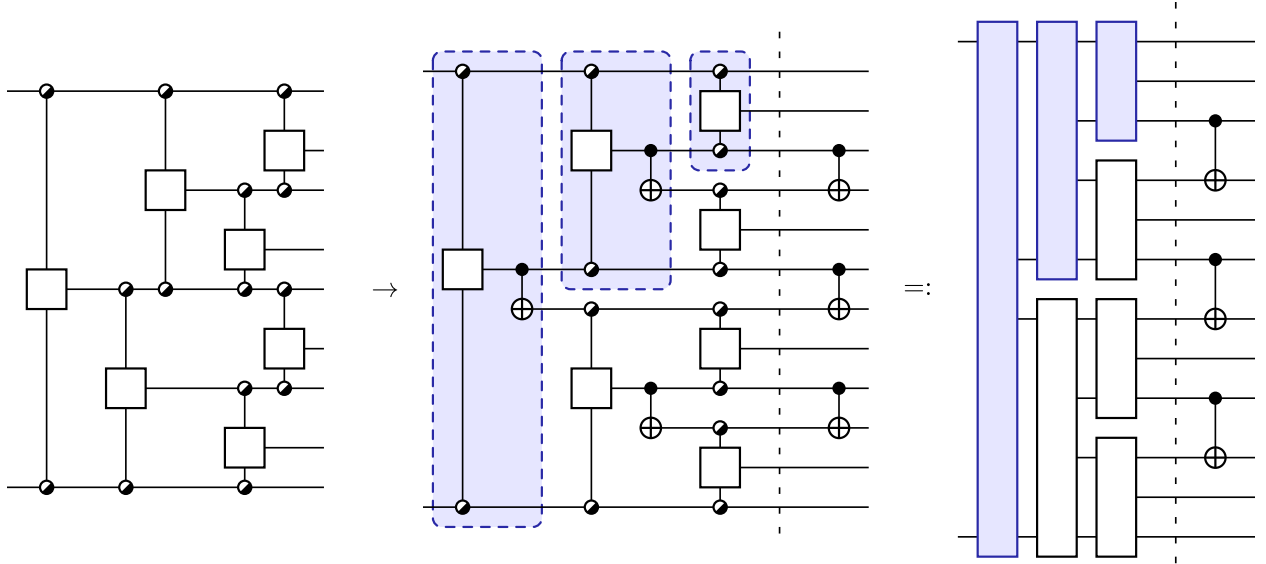


Figure 4: Determining the backward topology of the “ R stage” of the compressed Moore–Nilsson unitary. After some rewriting we obtain a binary tree with 2-qubit edges followed by a layer of CNOTs.

vertices at level s and level $s + 1$ of the tree there is a swap network permuting qubits to they are in the correct location for the next layer of gates. This swap network permutes qubits along the paths corresponding to embedded tree edges. The depth of this swap network is bounded by a constant factor times the maximum embedded edge length among the level s –($s + 1$) edges. The n -leaf binary tree has depth $\log n$, so the swap networks in total contribute $\mathcal{O}(d \log n)$ depth. The node layers in total contribute $\log n \cdot \mathcal{O}(1) = \mathcal{O}(\log n)$ depth, which is dominated by the swap network depth. $\diamond$

Combining these claims, we may appeal to minimax edge length embeddings of binary trees in 2D grids from the VLSI design literature [PRS81; RS81]. These results state a binary tree with n leaves may be embedded in a 2D grid with maximum edge length $\mathcal{O}(\sqrt{n}/\log n)$. This leads to a total depth of $\mathcal{O}(\sqrt{n})$ for implementing the first and last stages of a parallelized Moore–Nilsson circuit with a 2D architecture.

The central stage of the parallelized Moore–Nilsson circuit can be implemented in constant depth provided qubits on the 2D grid are arranged correctly.

Connecting these three stages requires at worst two arbitrary permutations on the grid of qubits, which take depth $\mathcal{O}(\sqrt{n})$ [ACG93]. Then the overall depth required to implement the parallelized Moore–Nilsson circuit is also $\mathcal{O}(\sqrt{n})$.

A straightforward lightcone argument shows that there are Moore–Nilsson unitaries that require depth $\Omega(\log n)$ in all-to-all connected circuits and depth $\Omega(\sqrt{n})$ in 2D circuits, so these upper bounds are asymptotically tight. $\square$

We close this section by noting that our proof of the second half of [Theorem 1.1](#) was somewhat wasteful in its use of ancilla qubits. With more careful bookkeeping it may be

possible to obtain a similar result using only $n + o(n)$ ancillae, but we leave this question to future work.

4 Discussion

This section covers two directions in which the results of this paper could be extended. In [Section 4.1](#) we discuss slight generalizations of Moore–Nilsson circuits on which our preprocessing techniques do not immediately apply. Further study of these circuits may lead to either new parallelization results, or super-logarithmic depth lower bounds. In [Section 4.2](#) we discuss how the results in this paper may be converted to practically-relevant compilation techniques, and highlight some initial progress in this direction given in [Appendix A](#).

4.1 Open problems near the Moore–Nilsson conjecture

Despite the progress made in this paper, we don’t know how to beat the naive depth bounds in any of the situations described below. More specifically, we don’t know how to compile the unitaries below to smaller depth than their naive implementations, either exactly or while retaining small error in operator norm. Interestingly, log-depth Frobenius norm approximations may be easier to obtain, but it is not clear if these can be strengthened to operator norm approximations. For many applications operator norm approximations seem necessary, such as when these circuits appear as subroutines in larger quantum algorithms.

Beyond one qubit of control

The techniques presented above appear to be specific to single-qubit controls from one unitary to the next. If several qubits from $U^{(i-1)}$ are used to control a $U^{(i)}$, then we do not know how to obtain any asymptotic depth reduction. This can be seen already from a staircase of *qutrit*-controlled unitaries. Attempts to extend the argument above to qutrits seem to require a 3×3 version of the CS decomposition, which is false in general. Parameter counting suggests this is not too surprising, but for completeness we include a concrete counterexample.

Proposition 4.1 (Counterexample to “ 3×3 CS decomposition”). *Let U be any unitary matrix proportional to a 6×6 matrix containing the following entries*

$$\left(\begin{array}{cc|cc} 1 & 0 & & \\ 0 & 2 & & \\ 1 & 2 & & \\ 0 & 3 & & \\ \hline & & * & \\ * & & & \end{array} \right),$$

Then there do not exist block-diagonal unitaries S and T and 2×2 diagonal matrices $\Sigma_{i,j}$, $i,j \in [3]$ such that

$$\begin{pmatrix} S_1 & 0 & 0 \\ 0 & S_2 & 0 \\ 0 & 0 & S_3 \end{pmatrix} U \begin{pmatrix} T_1 & 0 & 0 \\ 0 & T_2 & 0 \\ 0 & 0 & T_3 \end{pmatrix} = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} & \Sigma_{13} \\ \Sigma_{21} & \Sigma_{22} & \Sigma_{23} \\ \Sigma_{31} & \Sigma_{32} & \Sigma_{33} \end{pmatrix}. \quad (4.1)$$

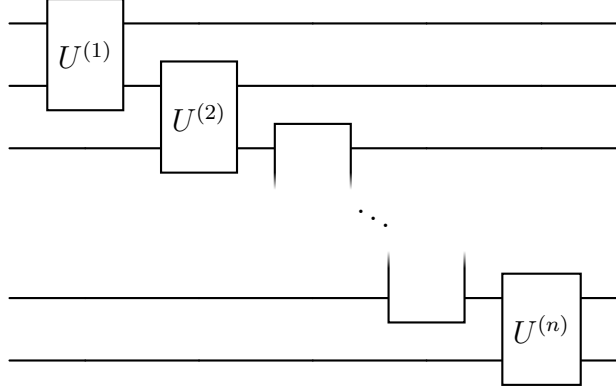


Figure 5: A Moore–Nilsson circuit variant with arbitrary two-qubit unitaries. It is an open question if circuits of this form are parallelizable.

Proof. Suppose we do have $S = S_1 \oplus S_2 \oplus S_3$ and $T = T_1 \oplus T_2 \oplus T_3$ achieving (4.1). Then

$$S_1 \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} T_1 = \Sigma_{11}$$

is diagonal, so

$$\Sigma_{11}^\dagger \Sigma_{11} = T_1^\dagger \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix} T_1 = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} 4 & 0 \\ 0 & 1 \end{pmatrix}$$

hence T_1 must be either diagonal or antidiagonal. Using (4.1) again,

$$S_2 \begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix} T_1 = \Sigma_{21}$$

is diagonal. Hence

$$\Sigma_{21}^\dagger \Sigma_{21} = \left(S_2 \begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix} T_1 \right)^\dagger S_2 \begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix} T_1 = T_1^\dagger \begin{pmatrix} 1 & 0 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix} T_1 = T_1^\dagger \begin{pmatrix} 1 & 2 \\ 2 & 13 \end{pmatrix} T_1$$

is diagonal too. However, this matrix is full since T_1 is a diagonal or antidiagonal unitary. $\square$

General unitaries

The techniques of this paper also do not apply to variants of Moore–Nilsson circuits where the controlled unitary operations are replaced by arbitrary two-qubit unitaries. (See Figure 5.)

Standard compilation techniques let us rewrite each two-qubit unitary in this cascade as a product of single-qubit rotations and two-qubit controlled unitaries. But it is unclear if the techniques in this paper can be extended to apply to circuits of this form.

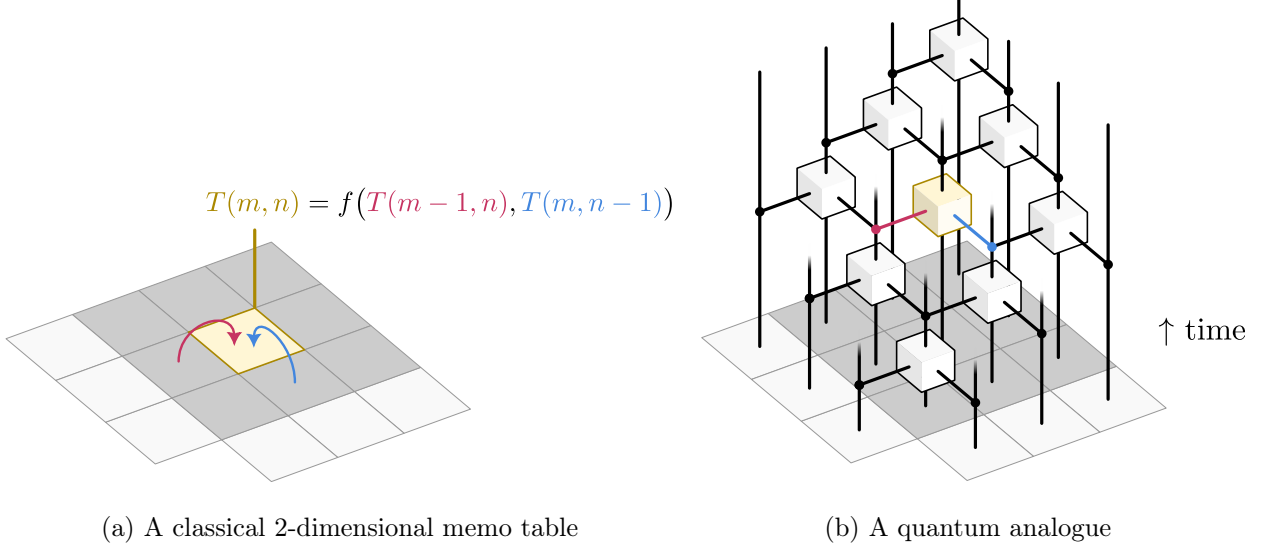


Figure 6: The current paper presents a quantum precomputation method for one-dimensional “quantum memo tables.” Does a quantum precomputation method exist for 2D tables and up?

Speedups for “quantum dynamic programming” in many dimensions?

The general parallelization techniques presented in this paper are specialized to one-dimensional “quantum memo tables,” or control cascade circuits. One can envision general k -dimensional quantum memo tables, in analogy with classical dynamic programming—see *e.g.*, Figure 6. Can we parallelize such circuits? Note that k -dimensional memo tables of maximum side-length n —as well as their quantum analogues—can be naively computed in circuit depth n (cells in the same diagonal are independent), so the appropriate goal is to asymptotically *beat* depth n . Classically this is possible with the Four-Russians method [Arl+70], down to depth $kn/\log(n)$.

4.2 Towards practical precomputation techniques

Beyond having complexity theoretic implications, are the circuit rewriting techniques described in this paper useful in practice? Answering this question likely requires understanding how these rewriting procedures affect other resource requirements of circuits, for example their T count. And this, in turn, requires a finer-grained investigation of the procedure, particularly the D , P and R matrices constructed in Lemmas 1.6 and 1.7.

Appendix A provides a first step in this direction, by giving inductive circuit decompositions of the S , T , and D unitaries generated by the CS decomposition of valley circuits. Beginning with these formula and then following the proofs outlined in this paper it should be possible to obtain clean descriptions of the circuits produced by parallelizing both Moore–Nilsson unitaries and other more-general unitary cascades of interest. This fine-grained investigation of circuits also has the potential to yield new asymptotic depth reductions for certain classes of unitary cascades, as it would allow replacing the worst-case bound of depth 4^k for implementing the P and multiplexer R gates with tighter upper bounds.

Finally, we mention the possibility that a closer investigation of the circuits produced by our parallelization procedure might reveal the opportunity to apply even more compilation techniques. As a concrete example, we mention that the parallelization of a Moore–Nilsson cascade of identical unitaries (for example, the $\mathcal{C}(H, H, \dots, H)$ circuit discussed in this paper’s introduction) produces columns of identical P and multiplexer R gates. Can the unitary “mass-production” theorems of [Kre22] be used in this setting?

References

- [ACG93] Noga Alon, Fan R. K. Chung, and Ronald L Graham. “Routing permutations on graphs via matchings”. In: *Proceedings of the twenty-fifth annual ACM symposium on Theory of Computing*. 1993, pp. 583–591 (cit. on p. 31).
- [AGS22] Atul Singh Arora, Alexandru Gheorghiu, and Uttam Singh. *Oracle separations of hybrid quantum-classical circuits*. 2022. arXiv: 2201.01904 [quant-ph]. URL: <https://arxiv.org/abs/2201.01904> (cit. on p. 13).
- [Arl+70] Vladimir L Arlazarov, E. A. Dinič, M. A. Kronrod, and IA Faradžev. “On economical construction of the transitive closure of a directed graph”. In: *Dokl. Akad. Nauk SSSR*. Vol. 194. 11. 1970, pp. 1209–1210 (cit. on pp. 1, 10, 11, 34).
- [Aro+23] Atul Singh Arora, Andrea Coladangelo, Matthew Coudron, Alexandru Gheorghiu, Uttam Singh, and Hendrik Waldner. “Quantum Depth in the Random Oracle Model”. In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*. STOC 2023. Orlando, FL, USA: Association for Computing Machinery, 2023, pp. 1111–1124. ISBN: 9781450399135. DOI: 10.1145/3564246.3585153. URL: <https://doi.org/10.1145/3564246.3585153> (cit. on p. 13).
- [Ble90] Guy E. Blelloch. “Prefix sums and their applications”. In: (1990) (cit. on p. 4).
- [BM03] Stephen S. Bullock and Igor L. Markov. “An arbitrary twoqubit computation in 23 elementary gates or less”. In: *Proceedings of the 40th Annual Design Automation Conference*. 2003, pp. 324–329 (cit. on p. 5).
- [BM04] Stephen S. Bullock and Igor L. Markov. “Asymptotically optimal circuits for arbitrary n -qubit diagonal computations”. In: *Quantum Info. Comput.* 4.1 (Jan. 2004), pp. 27–47. ISSN: 1533-7146 (cit. on pp. 9, 16, 20).
- [BN00] Dan Boneh and Moni Naor. “Timed Commitments”. In: *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*. Ed. by Mihir Bellare. Vol. 1880. Lecture Notes in Computer Science. Springer, 2000, pp. 236–254. DOI: 10.1007/3-540-44598-6_15. URL: https://doi.org/10.1007/3-540-44598-6_15 (cit. on p. 3).

- [Bon+25] Joseph Bonneau, Benedikt Bünz, Miranda Christ, and Yuval Efron. “Good Things Come to Those Who Wait: Dishonest-Majority Coin-Flipping Requires Delay Functions”. In: *Advances in Cryptology – EUROCRYPT 2025: 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4–8, 2025, Proceedings, Part VII*. Madrid, Spain: Springer-Verlag, 2025, pp. 225–253. ISBN: 978-3-031-91097-5. DOI: [10.1007/978-3-031-91098-2_9](https://doi.org/10.1007/978-3-031-91098-2_9). URL: https://doi.org/10.1007/978-3-031-91098-2_9 (cit. on p. 3).
- [Bre+02] Michael J. Bremner, Christopher M. Dawson, Jennifer L. Dodd, Alexei Gilchrist, Aram W. Harrow, Duncan Mortimer, Michael A. Nielsen, and Tobias J Osborne. “Practical scheme for quantum computation with any two-qubit entangling gate”. In: *Physical review letters* 89.24 (2002), p. 247902 (cit. on p. 5).
- [Coo81] Stephen A. Cook. “Towards a complexity theory of synchronous parallel computation”. In: *Enseign. Math.* (2) 27.1-2 (1981), pp. 99–124. ISSN: 0013-8584 (cit. on p. 3).
- [CRR21] Geoffroy Couteau, A. W. Roscoe, and Peter Y. A. Ryan. “Partially-Fair Computation from Timed-Release Encryption and Oblivious Transfer”. In: *Information Security and Privacy - 26th Australasian Conference, ACISP 2021, Virtual Event, December 1-3, 2021, Proceedings*. Ed. by Joonsang Baek and Sushmita Ruj. Vol. 13083. Lecture Notes in Computer Science. Springer, 2021, pp. 330–349. DOI: [10.1007/978-3-030-90567-5_17](https://doi.org/10.1007/978-3-030-90567-5_17). URL: https://doi.org/10.1007/978-3-030-90567-5_17 (cit. on p. 3).
- [CV89] Richard Cole and Uzi Vishkin. “Faster optimal parallel prefix sums and list ranking”. In: *Information and computation* 81.3 (1989), pp. 334–352 (cit. on p. 4).
- [CW00] Richard Cleve and John Watrous. “Fast parallel circuits for the quantum Fourier transform”. In: *Proceedings 41st Annual Symposium on Foundations of Computer Science*. IEEE. 2000, pp. 526–536 (cit. on p. 18).
- [DK69] Chandler Davis and William Kahan. “Some new bounds on perturbation of subspaces”. In: *Bulletin of the American Mathematical Society* 75 (1969), pp. 863–868. URL: <https://api.semanticscholar.org/CorpusID:122093640> (cit. on p. 14).
- [GNS18] Evan S. Gawlik, Yuji Nakatsukasa, and Brian D. Sutton. “A backward stable algorithm for computing the CS decomposition via the polar decomposition”. In: *SIAM J. Matrix Anal. Appl.* 39.3 (2018), pp. 1448–1469. ISSN: 0895-4798,1095-7162. DOI: [10.1137/18M1182747](https://doi.org/10.1137/18M1182747). URL: <https://doi.org/10.1137/18M1182747> (cit. on p. 12).
- [Gus97] Dan Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, May 1997. ISBN: 9780511574931. DOI: [10.1017/cbo9780511574931](https://doi.org/10.1017/cbo9780511574931). URL: <http://dx.doi.org/10.1017/CBO9780511574931> (cit. on p. 10).

- [Hås93] Johan Håstad. “The Shrinkage Exponent is 2”. In: *Proceedings of the 34th Annual ACM/IEEE Symposium on Foundations of Computer Science (FOCS 1993)*. IEEE, 1993, pp. 114–123. DOI: [10.1109/SFCS.1993.366876](https://doi.org/10.1109/SFCS.1993.366876) (cit. on p. 3).
- [Hås98] Johan Håstad. “The Shrinkage Exponent of de Morgan Formulas is 2”. In: *SIAM Journal on Computing* 27.1 (1998), pp. 48–64. DOI: [10.1137/S0097539794261556](https://doi.org/10.1137/S0097539794261556). eprint: <https://doi.org/10.1137/S0097539794261556>. URL: <https://doi.org/10.1137/S0097539794261556> (cit. on p. 3).
- [Jor75] Camille Jordan. “Essai sur la géométrie à n dimensions”. fr. In: *Bulletin de la Société Mathématique de France* 3 (1875), pp. 103–174. DOI: [10.24033/bsmf.90](https://doi.org/10.24033/bsmf.90). URL: <https://www.numdam.org/articles/10.24033/bsmf.90/> (cit. on p. 14).
- [Kah+25] Gregory D Kahanamoku-Meyer, John Blue, Thiago Bergamaschi, Craig Gidney, and Isaac L Chuang. “A log-depth in-place quantum Fourier transform that rarely needs ancillas”. In: *arXiv preprint arXiv:2505.00701* (2025) (cit. on p. 12).
- [Kre+23] William Kretschmer, Luowen Qian, Makrand Sinha, and Avishay Tal. “Quantum cryptography in algorithmica”. In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*. 2023, pp. 1589–1602 (cit. on p. 5).
- [Kre21] William Kretschmer. “Quantum Pseudorandomness and Classical Complexity”. In: *16th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2021 (cit. on p. 5).
- [Kre22] William Kretschmer. “Quantum Mass Production Theorems”. In: *arXiv preprint arXiv:2212.14399* (2022) (cit. on p. 35).
- [KRW95] Mauricio Karchmer, Ran Raz, and Avi Wigderson. “Super-Logarithmic Depth Lower Bounds Via the Direct Sum in Communication Complexity”. In: *Comput. Complex.* 5.3/4 (1995), pp. 191–204. DOI: [10.1007/BF01206317](https://doi.org/10.1007/BF01206317). URL: <https://doi.org/10.1007/BF01206317> (cit. on p. 3).
- [LMW24] Alex Lombardi, Fermi Ma, and John Wright. “A One-Query Lower Bound for Unitary Synthesis and Breaking Quantum Cryptography”. In: *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*. Ed. by Bojan Mohar, Igor Shinkar, and Ryan O’Donnell. ACM, 2024, pp. 979–990. DOI: [10.1145/3618260.3649650](https://doi.org/10.1145/3618260.3649650). URL: <https://doi.org/10.1145/3618260.3649650> (cit. on p. 5).
- [Mei25] Or Meir. “Toward Better Depth Lower Bounds: A KRW-like Theorem For Strong Composition”. In: *SIAM Journal on Computing* 54.5 (2025), pp. 1193–1240. DOI: [10.1137/23M158615X](https://doi.org/10.1137/23M158615X). eprint: <https://doi.org/10.1137/23M158615X>. URL: <https://doi.org/10.1137/23M158615X> (cit. on p. 3).
- [MN01] Cristopher Moore and Martin Nilsson. “Parallel Quantum Computation and Quantum Codes”. In: *SIAM Journal on Computing* 31.3 (2001), pp. 799–815. DOI: [10.1137/S0097539799355053](https://doi.org/10.1137/S0097539799355053). eprint: <https://doi.org/10.1137/S0097539799355053>. URL: <https://doi.org/10.1137/S0097539799355053> (cit. on pp. 1, 4).

- [Möt+04] Mikko Möttönen, Juha J. Vartiainen, Ville Bergholm, and Martti M. Salomaa. “Quantum Circuits for General Multiqubit Gates”. In: *Phys. Rev. Lett.* 93 (13 Sept. 2004), p. 130502. DOI: [10.1103/PhysRevLett.93.130502](https://link.aps.org/doi/10.1103/PhysRevLett.93.130502). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.93.130502> (cit. on pp. 7, 8).
- [PRS81] M. S. Paterson, W. L. Ruzzo, and L. Snyder. “Bounds on minimax edge length for complete binary trees”. In: *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*. STOC ’81. Milwaukee, Wisconsin, USA: Association for Computing Machinery, 1981, pp. 293–299. ISBN: 9781450373920. DOI: [10.1145/800076.802481](https://doi.org/10.1145/800076.802481). URL: <https://doi.org/10.1145/800076.802481> (cit. on pp. 11, 31).
- [PW94] C. C. Paige and M. Wei. “History and generality of the CS decomposition”. In: *Linear Algebra and its Applications* 208-209 (1994), pp. 303–326. ISSN: 0024-3795. DOI: [https://doi.org/10.1016/0024-3795\(94\)90446-4](https://doi.org/10.1016/0024-3795(94)90446-4). URL: <https://www.sciencedirect.com/science/article/pii/0024379594904464> (cit. on pp. 8, 14).
- [RS81] Walter L. Ruzzo and Lawrence Snyder. “Minimum Edge Length Planar Embeddings of Trees”. In: *VLSI Systems and Computations*. Ed. by H. T. Kung, Bob Sproull, and Guy Steele. Berlin, Heidelberg: Springer Berlin Heidelberg, 1981, pp. 119–123. ISBN: 978-3-642-68402-9. DOI: [10.1007/978-3-642-68402-9_14](https://doi.org/10.1007/978-3-642-68402-9_14). URL: https://doi.org/10.1007/978-3-642-68402-9_14 (cit. on pp. 11, 31).
- [RSW96] Ronald L. Rivest, Adi Shamir, and David A. Wagner. *Time-Lock Puzzles and Timed-Release Crypto*. Technical Report MIT/LCS/TR-684. Revised March 10, 1996. Cambridge, MA, USA: MIT Laboratory for Computer Science, Feb. 1996. URL: <https://people.csail.mit.edu/rivest/pubs/RSW96.pdf> (cit. on p. 3).
- [Son+25] Jeongrak Son, Marek Gluza, Ryuji Takagi, and Nelly H. Y. Ng. “Quantum Dynamic Programming”. In: *Phys. Rev. Lett.* 134 (18 May 2025), p. 180602. DOI: [10.1103/PhysRevLett.134.180602](https://link.aps.org/doi/10.1103/PhysRevLett.134.180602). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.134.180602> (cit. on p. 12).
- [TT24] Ewin Tang and Kevin Tian. “A CS guide to the quantum singular value transformation”. In: *2024 Symposium on Simplicity in Algorithms, SOSA 2024, Alexandria, VA, USA, January 8-10, 2024*. Ed. by Merav Parter and Seth Pettie. SIAM, 2024, pp. 121–143. DOI: [10.1137/1.9781611977936.13](https://doi.org/10.1137/1.9781611977936.13). URL: <https://doi.org/10.1137/1.9781611977936.13> (cit. on p. 8).
- [Tuc99] Robert R. Tucci. *A Rudimentary Quantum Compiler(2cnd Ed.)* 1999. arXiv: [quant-ph/9902062](https://arxiv.org/abs/quant-ph/9902062) [quant-ph] (cit. on pp. 7–9, 16, 29).
- [VMS04] Juha J. Vartiainen, Mikko Möttönen, and Martti M. Salomaa. “Efficient Decomposition of Quantum Gates”. In: *Phys. Rev. Lett.* 92 (17 Apr. 2004), p. 177902. DOI: [10.1103/PhysRevLett.92.177902](https://link.aps.org/doi/10.1103/PhysRevLett.92.177902). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.92.177902> (cit. on pp. 9, 16, 29).

A Explicit formulas for the CS decomposition of a valley circuit

The CS decomposition of a valley circuit can be written in terms of the CS decomposition of the unitaries which define the valley circuit. In [Lemma 3.2](#) we provided an algebraic formula for the stubs of the valley circuit. In this section, we provide formulas for every component of a CS Decomposition for a basic valley; here, we do it totally in terms of circuit diagrams.

We start by reviewing CS decomposition notation. Then we state the main result [Proposition A.1](#).

A.1 Notation for this appendix

We begin with a remark on notation, so that the forthcoming set of formulas is consistent with the ones which came before. In this section, when we take the CS decomposition of a circuit Ξ , instead of using

$$\Xi = \begin{array}{c} \text{---} \bullet \text{---} \boxed{\Phi^\dagger} \text{---} \boxed{D^\Xi} \text{---} \boxed{\Phi} \text{---} \bullet \text{---} \\ | \quad \quad \quad | \quad \quad \quad | \\ \boxed{S^\Xi} \text{---} \text{---} \text{---} \boxed{T^\Xi} \end{array} \quad (\text{A.1})$$

we will use a shorthand version

$$\Xi = \begin{array}{c} \text{---} \bullet \text{---} \boxed{\Sigma^\Xi} \text{---} \bullet \text{---} \\ | \quad \quad \quad | \\ \boxed{S^\Xi} \text{---} \text{---} \text{---} \boxed{T^\Xi} \end{array} \quad (\text{A.2})$$

where we take

$$\Sigma^\Xi = \begin{array}{c} \boxed{\Phi^\dagger} \text{---} \boxed{D^\Xi} \text{---} \boxed{\Phi} \\ | \quad \quad \quad | \\ \text{---} \text{---} \text{---} \end{array} = \begin{pmatrix} \Sigma_1^\Xi & \Sigma_2^\Xi \\ -\Sigma_2^\Xi & \Sigma_1^\Xi \end{pmatrix} \quad (\text{A.3})$$

where Σ_1^Ξ and Σ_2^Ξ are nonnegative diagonal matrices.

A.2 The formula for the CS Decomposition of a valley

Proposition A.1. *Given a valley W of U and V in accordance with [Figure 7](#) and CS Decompositions of U and V , a CS Decomposition of W*

$$W = \begin{array}{c} \text{---} \bullet \text{---} \boxed{\Sigma^W} \text{---} \bullet \text{---} \\ | \quad \quad \quad | \\ \boxed{S^W} \text{---} \text{---} \text{---} \boxed{T^W} \end{array} \quad (\text{A.4})$$

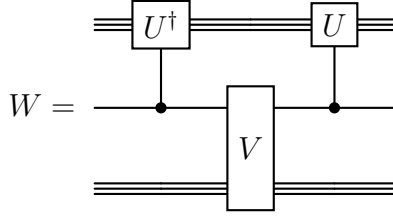
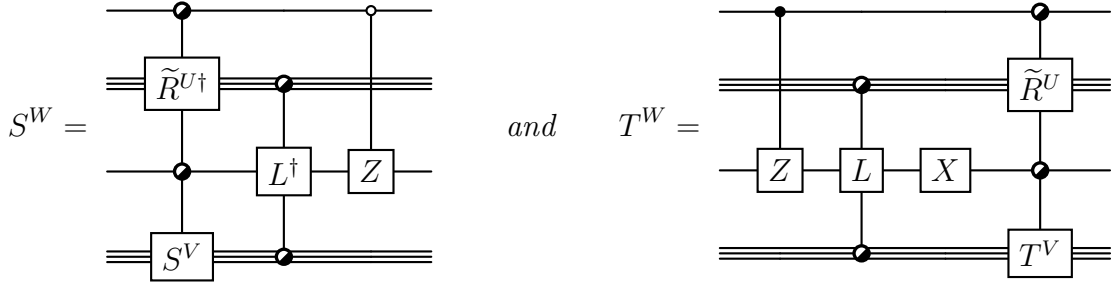


Figure 7: The basic valley

has components

$$\Sigma^W := \begin{pmatrix} \Sigma_1^W & \Sigma_2^W \\ -\Sigma_2^W & \Sigma_1^W \end{pmatrix} \quad \text{with} \quad \Sigma_2^W := \begin{array}{c} \equiv \equiv \\ \boxed{\Sigma_2^U} \\ \equiv \equiv \\ \text{---} \\ \equiv \equiv \\ \boxed{\Sigma_2^V} \\ \equiv \equiv \end{array} \quad \text{and} \quad \Sigma_1^W := \sqrt{I - (\Sigma_2^W)^2}$$



where $\tilde{R}^U$ is represented by

$$\tilde{R}^U = \begin{array}{c} \text{---} \\ \bullet \\ \equiv \equiv \\ \boxed{\tilde{R}^U} \\ \equiv \equiv \\ \bullet \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \bullet \\ \equiv \equiv \\ \boxed{S^{U\dagger}} \quad \boxed{T^U} \\ \equiv \equiv \\ \bullet \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \bullet \\ \equiv \equiv \\ \boxed{S_0^{U\dagger}} \quad \boxed{S_1^{U\dagger}} \quad \boxed{T_0^U} \quad \boxed{T_1^U} \\ \equiv \equiv \\ \bullet \\ \text{---} \end{array} \quad (\text{A.5})$$

and L represents a list of $(\dim W)/4$ one-qubit unitaries are defined in [Equation \(A.20\)](#).

Note: $\tilde{R}$ is the same as R used in the parallelization identity with reversed qubits.

The proof is based on two lemmas each presented in its own subsubsection.

A.2.1 Reduction from a general valley to a valley of diagonal matrices

Lemma A.2. Given CS decompositions of U and V

$$U = \begin{array}{c} \text{---} \\ \bullet \\ \equiv \equiv \\ \boxed{S^U} \quad \boxed{\Sigma^U} \quad \boxed{T^U} \\ \equiv \equiv \\ \bullet \\ \text{---} \end{array} \quad \text{and} \quad V = \begin{array}{c} \text{---} \\ \bullet \\ \equiv \equiv \\ \boxed{S^V} \quad \boxed{\Sigma^V} \quad \boxed{T^V} \\ \equiv \equiv \\ \bullet \\ \text{---} \end{array}. \quad (\text{A.6})$$

[illegible]
$$\begin{array}{c} \bullet \\ \text{---} \\ \boxed{\widetilde{R}^U} \\ \text{---} \\ \bullet \end{array} = \begin{array}{c} \bullet \quad \bullet \\ \text{---} \quad \text{---} \\ \boxed{S^{U\dagger}} \quad \boxed{T^U} \\ \text{---} \quad \text{---} \\ \circ \quad \bullet \end{array} = \begin{array}{c} \circ \quad \bullet \quad \circ \quad \bullet \\ \text{---} \quad \text{---} \quad \text{---} \quad \text{---} \\ \boxed{S_0^{U\dagger}} \quad \boxed{S_1^{U\dagger}} \quad \boxed{T_0^U} \quad \boxed{T_1^U} \\ \text{---} \quad \text{---} \quad \text{---} \quad \text{---} \\ \circ \quad \circ \quad \bullet \quad \bullet \end{array} \quad (\text{A.8})$$

$W =$

n multiply by

 which is the identity on both the left and right

then commute them through the closed controls to get

$$W = \begin{array}{c} \text{Diagram (A.9): A quantum circuit with three horizontal lines. The top line has control dots for gates S^U, $T^{U\dagger}$, $\Sigma^{U\dagger}$, $S^{U\dagger}$, S^U, Σ^U, $S^{U\dagger}$, and T^U. The middle line has control dots for gates S^U, $T^{U\dagger}$, $\Sigma^{U\dagger}$, Σ^U, $S^{U\dagger}$, and T^U. The bottom line has control dots for gates S^U, $T^{U\dagger}$, $\Sigma^{U\dagger}$, Σ^U, $S^{U\dagger}$, and T^U. The gates are arranged in a sequence: S^U and $T^{U\dagger}$ on the top and middle lines, followed by $\Sigma^{U\dagger}$ on the top line, then $S^{U\dagger}$ on the top line, then a long horizontal line, then S^U on the top line, followed by Σ^U on the top line, then $S^{U\dagger}$ on the top line, and finally T^U on the top line. The bottom line has gates S^V and T^V in the middle, with Σ^V in between. The bottom line is represented by multiple parallel lines.$$
(A.9)

The center S^U terms cancel, the S^V and T^V terms commute out, and the S^U and T^U terms combine into $\tilde{R}^U$ to get

$$W = \begin{array}{c} \text{Diagram (A.10): A simplified quantum circuit. The top line has control dots for gates $\tilde{R}^{U\dagger}$, $\Sigma^{U\dagger}$, Σ^U, and $\tilde{R}^U$. The middle line has control dots for gates $\tilde{R}^{U\dagger}$, $\Sigma^{U\dagger}$, Σ^U, and $\tilde{R}^U$. The bottom line has control dots for gates $\tilde{R}^{U\dagger}$, $\Sigma^{U\dagger}$, Σ^U, and $\tilde{R}^U$. The gates are arranged in a sequence: $\tilde{R}^{U\dagger}$ on the top and middle lines, followed by $\Sigma^{U\dagger}$ on the top line, then Σ^U on the top line, and finally $\tilde{R}^U$ on the top line. The bottom line has gates S^V and T^V in the middle, with Σ^V in between. The bottom line is represented by multiple parallel lines.$$
(A.10)

which confirms the factorization. □

A.2.2 Valleys of diagonal matrices

Lemma A.3. Let $\Sigma^U = \begin{pmatrix} \Sigma_1^U & \Sigma_2^U \\ -\Sigma_2^U & \Sigma_1^U \end{pmatrix}$ be a unitary circuit where Σ_1^U and Σ_2^U are diagonal and let Σ^V be similarly defined. Given a valley circuit W^Σ of Σ^U and Σ^V in accordance with

$$W^\Sigma = \begin{array}{c} \text{Diagram (A.11): A quantum circuit with three horizontal lines. The top line has control dots for gates $\Sigma^{U\dagger}$ and Σ^U. The middle line has control dots for gates $\Sigma^{U\dagger}$ and Σ^U. The bottom line has control dots for gates $\Sigma^{U\dagger}$ and Σ^U. The gates are arranged in a sequence: $\Sigma^{U\dagger}$ on the top and middle lines, followed by Σ^U on the top line, then Σ^V on the bottom line, and finally Σ^U on the top line. The bottom line is represented by multiple parallel lines.$$
(A.11)

a CS Decomposition of W^Σ is given by

$$\Sigma_2^{W^\Sigma} := \begin{array}{c} \text{---} \Sigma_2^U \text{---} \\ \text{---} \Sigma_2^V \text{---} \end{array} \quad \text{and} \quad \Sigma_1^{W^\Sigma} := \sqrt{I - (\Sigma_2^{W^\Sigma})^2}$$

$$S^{W^\Sigma} = \begin{array}{c} \text{---} \text{---} \\ \bullet \text{---} \text{---} \\ \text{---} L^\dagger \text{---} Z \text{---} \\ \bullet \text{---} \text{---} \end{array} \quad \text{and} \quad T^{W^\Sigma} = \begin{array}{c} \text{---} \text{---} \\ \bullet \text{---} \text{---} \\ \text{---} Z \text{---} L \text{---} X \text{---} \\ \bullet \text{---} \text{---} \end{array}$$

where L is a list of $\dim W/4$ one-qubit unitaries.

Proof. We write W^Σ as a block 2×2 matrix, based on inputs and outputs on the middle wire. This allows us to write W^Σ as the sum of the 4 blocks. We will use $|input\rangle\langle output|$ on the middle wire to denote which block is selected. The sum of the four blocks can be written:

$$W^\Sigma = \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} |0\rangle\langle 0| \text{---} \\ \text{---} \Sigma_1^V \text{---} \end{array} + \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} |1\rangle\langle 1| \text{---} \\ \text{---} \Sigma_1^V \text{---} \end{array} + \begin{array}{c} \text{---} \Sigma^U \text{---} \\ \text{---} |0\rangle\langle 1| \text{---} \\ \text{---} \Sigma_2^V \text{---} \end{array} + \begin{array}{c} \text{---} \Sigma^{U\dagger} \text{---} \\ \text{---} |1\rangle\langle 0| \text{---} \\ \text{---} -\Sigma_2^V \text{---} \end{array}$$

We next consider the blocks of W^Σ individually. First, we start by analyzing the off-diagonal blocks.

Off-diagonal blocks. W_{01}^Σ is the off-diagonal block where we take the input $\langle 0|$ and the output $|1\rangle$ on the top wire. With the identity on the top wire, the first two terms of W_{01}^Σ are 0 and we are left with

$$W_{01}^\Sigma = \begin{array}{c} \langle 0| \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} W^\Sigma \begin{array}{c} \text{---} |1\rangle \\ \text{---} \\ \text{---} \\ \text{---} \end{array} = \begin{array}{c} \text{---} \Sigma_2^U \text{---} \\ \text{---} |0\rangle\langle 1| \text{---} \\ \text{---} \Sigma_2^V \text{---} \end{array} + \begin{array}{c} \text{---} -\Sigma_2^U \text{---} \\ \text{---} |1\rangle\langle 0| \text{---} \\ \text{---} -\Sigma_2^V \text{---} \end{array} = \begin{array}{c} \text{---} \Sigma_2^U \text{---} \\ \text{---} X \text{---} \\ \text{---} \Sigma_2^V \text{---} \end{array} = \begin{array}{c} \text{---} \Sigma_2^{W^\Sigma} \text{---} \\ \text{---} X \text{---} \\ \text{---} \end{array}$$

because $|0\rangle\langle 1| + |1\rangle\langle 0| = X$.

We can perform similar operations to find W_{10}^Z

[illegible]

Diagonal blocks. Next, we consider the W_{00}^Σ block. We have

$$\begin{aligned}
W_{00}^\Sigma = & \text{---} \boxed{|0\rangle\langle 0|} \text{---} + \text{---} \boxed{|1\rangle\langle 1|} \text{---} + \text{---} \boxed{|0\rangle\langle 1|} \text{---} + \text{---} \boxed{|1\rangle\langle 0|} \text{---} \\
& \text{---} \boxed{\Sigma_1^V} \text{---} \quad \text{---} \boxed{\Sigma_1^V} \text{---} \quad \text{---} \boxed{\Sigma_2^V} \text{---} \quad \text{---} \boxed{-\Sigma_2^V} \text{---}
\end{aligned} \tag{A.12}$$

We can combine the first two terms as $|0\rangle\langle 0| + |1\rangle\langle 1| = I_2$ and we can combine the second two terms as $|0\rangle\langle 1| - |1\rangle\langle 0| = ZX$. W_{00}^Σ can then be written as

$$W_{00}^\Sigma = \text{---} + \text{---} \boxed{ZX} \text{---} \quad (\text{A.13})$$

We observe that W_{00}^Σ is diagonal except on the middle wire. If we were to swap the middle and bottom wires using P_{swap} , we would have

$$P_{swap}^\dagger W_{00}^\Sigma P_{swap} = \text{diagram with } \Sigma_1^V \text{ box} + \text{diagram with } \Sigma_2^V \text{ box} . \quad (\text{A.14})$$

The matrix representing this circuit is the sum of two matrices. The first circuit is $I \otimes \Sigma_1^V \otimes I$ which is block diagonal with the blocks of the form $a_j I_2$ with a_j a real number. The second circuit is $\Sigma_1^U \otimes \Sigma_2^V \otimes \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ which is block diagonal with the blocks of the form $b_j \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ with b_j a real number.

Combining these, $P_{\text{swap}}^\dagger W_{00}^\Sigma P_{\text{swap}}$ is a block diagonal matrix with blocks

$$B_j = \begin{pmatrix} a_j & b_j \\ -b_j & a_j \end{pmatrix} \quad (\text{A.15})$$

With this, we can replace the top sets of wires with a large set of controls:

$$P_{\text{swap}}^\dagger W_{00}^\Sigma P_{\text{swap}} = \begin{array}{c} \text{---} \bullet \text{---} \\ | \\ \text{---} \bullet \text{---} \\ | \\ \text{---} [B] \text{---} \end{array} . \quad (\text{A.16})$$

Reversing the permutations gives us the result that we can write W_{00}^Σ as a doubly multi-controlled list of one-qubit unitaries:

$$W_{00}^\Sigma = \begin{array}{c} \text{---} \bullet \text{---} \\ | \\ \text{---} [B] \text{---} \\ | \\ \text{---} \bullet \text{---} \end{array} . \quad (\text{A.17})$$

We then apply the X gate that converts W_{01}^Σ and W_{10}^Σ into Σ_2^W , and then define B^W as

$$\begin{array}{c} \text{---} \bullet \text{---} \\ | \\ \text{---} [B^W] \text{---} \\ | \\ \text{---} \bullet \text{---} \end{array} := \begin{array}{c} \text{---} \bullet \text{---} \\ | \\ \text{---} [B] [X] \text{---} \\ | \\ \text{---} \bullet \text{---} \end{array} = \begin{array}{c} \text{---} \\ | \\ \text{---} [W_{00}^\Sigma] [X] \text{---} \\ | \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ | \\ \text{---} [X] \text{---} \\ | \\ \text{---} \end{array} + \begin{array}{c} \text{---} \\ | \\ \text{---} [Z] \text{---} \\ | \\ \text{---} \end{array} = \begin{array}{c} \text{---} [\Sigma_1^U] \text{---} \\ | \\ \text{---} [\Sigma_1^V] \text{---} \\ | \\ \text{---} [\Sigma_2^V] \text{---} \end{array} \quad (\text{A.18})$$

where $B^W := (B_1^W, B_2^W, \dots, B_{\dim(W)/4}^W)$ with

$$B_j^W = \begin{pmatrix} b_j & a_j \\ a_j & -b_j \end{pmatrix}.$$

The matrix B_j^W has eigenvalues are c_j and $-c_j$ where $c_j = \sqrt{a_j^2 + b_j^2}$. We let L_j denote a unitary matrix diagonalizing B_j^W , namely,

$$B_j^W = L_j^\dagger \begin{pmatrix} c_j & 0 \\ 0 & -c_j \end{pmatrix} L_j = L_j^\dagger c_j Z L_j \quad (\text{A.19})$$

Form the list $L = (L_1, \dots, L_{\dim(W)/4})$ and use it in the circuit

$$\begin{array}{c} \text{---} \bullet \text{---} \\ | \\ \text{---} [B^W] \text{---} \\ | \\ \text{---} \bullet \text{---} \end{array} = \begin{array}{c} \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \\ | \quad | \quad | \\ \text{---} [L^\dagger] [C] [Z] [L] \text{---} \\ | \quad | \quad | \\ \text{---} \bullet \text{---} \bullet \text{---} \bullet \text{---} \end{array} \quad (\text{A.20})$$

where $C := (c_1 I, c_2 I, \dots, c_{\dim(W)/4} I)$. Notably, C ends up being a real diagonal matrix with entries equal to the singular values of W_{00}^Σ . Since W is unitary and W_{01}^Σ times X on the middle wire is equal to $\Sigma_2^{W^\Sigma}$ and $\Sigma_1^{W^\Sigma} = \sqrt{1 - (\Sigma_2^{W^\Sigma})^2}$, this means that C has the same singular values as W_{00}^Σ in the same order as $\Sigma_1^{W^\Sigma}$, and

$$\begin{array}{c} \text{---} \bullet \text{---} \\ | \\ \boxed{C} \\ | \\ \text{---} \bullet \text{---} \end{array} = \Sigma_1^{W^\Sigma} \quad (\text{A.21})$$

so

$$\begin{array}{c} \text{---} \bullet \text{---} \\ | \\ \boxed{B^W} \\ | \\ \text{---} \bullet \text{---} \end{array} = \begin{array}{c} \text{---} \bullet \text{---} \\ | \\ \boxed{L^\dagger} \\ | \\ \text{---} \bullet \text{---} \end{array} \Sigma_1^{W^\Sigma} \begin{array}{c} \text{---} \bullet \text{---} \\ | \\ \boxed{Z} \\ | \\ \text{---} \bullet \text{---} \end{array} \begin{array}{c} \text{---} \bullet \text{---} \\ | \\ \boxed{L} \\ | \\ \text{---} \bullet \text{---} \end{array} \quad (\text{A.22})$$

The computations above give us the recipes needed to evaluate the CS Decomposition stated in the lemma to verify that it is indeed W^Σ . We present this next.

Evaluating the CS decomposition. The CS decomposition expands to

$$\begin{array}{c} \text{---} \circ \text{---} \\ | \\ \text{---} \bullet \text{---} \\ | \\ \boxed{L^\dagger} \quad \boxed{Z} \\ | \\ \text{---} \bullet \text{---} \end{array} \Sigma_1^{W^\Sigma} \begin{array}{c} \text{---} \bullet \text{---} \\ | \\ \boxed{Z} \quad \boxed{L} \quad \boxed{X} \\ | \\ \text{---} \bullet \text{---} \end{array} \quad (\text{A.23})$$

We write the CS decomposition as a block 2×2 matrix, based on inputs and output of the

top wire to get

$$\begin{aligned}
 & \begin{array}{c} \text{---} |0\rangle\langle 0| \text{---} \\ \text{---} |1\rangle\langle 1| \text{---} \end{array} \\
 = & \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array} \begin{array}{c} L^\dagger \\ L^\dagger \\ L^\dagger \end{array} \begin{array}{c} Z \\ Z \\ Z \end{array} \begin{array}{c} \Sigma_1^{W^\Sigma} \\ \Sigma_1^{W^\Sigma} \\ \Sigma_1^{W^\Sigma} \end{array} \begin{array}{c} L \\ L \\ L \end{array} \begin{array}{c} X \\ X \\ X \end{array} \begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array} \text{---} \\
 + & \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array} \begin{array}{c} L^\dagger \\ L^\dagger \\ L^\dagger \end{array} \begin{array}{c} \Sigma_1^{W^\Sigma} \\ \Sigma_1^{W^\Sigma} \\ \Sigma_1^{W^\Sigma} \end{array} \begin{array}{c} Z \\ Z \\ Z \end{array} \begin{array}{c} L \\ L \\ L \end{array} \begin{array}{c} X \\ X \\ X \end{array} \begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array} \text{---}
 \end{aligned} \tag{A.24}$$

$$\begin{aligned}
 & \begin{array}{c} \text{---} |1\rangle\langle 0| \text{---} \\ \text{---} |0\rangle\langle 1| \text{---} \end{array} \\
 + & \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array} \begin{array}{c} L^\dagger \\ L^\dagger \\ L^\dagger \end{array} \begin{array}{c} -\Sigma_2^{W^\Sigma} \\ -\Sigma_2^{W^\Sigma} \\ -\Sigma_2^{W^\Sigma} \end{array} \begin{array}{c} L \\ L \\ L \end{array} \begin{array}{c} X \\ X \\ X \end{array} \begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array} \text{---} \\
 + & \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array} \begin{array}{c} L^\dagger \\ L^\dagger \\ L^\dagger \end{array} \begin{array}{c} Z \\ Z \\ Z \end{array} \begin{array}{c} \Sigma_2^{W^\Sigma} \\ \Sigma_2^{W^\Sigma} \\ \Sigma_2^{W^\Sigma} \end{array} \begin{array}{c} Z \\ Z \\ Z \end{array} \begin{array}{c} L \\ L \\ L \end{array} \begin{array}{c} X \\ X \\ X \end{array} \begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array} \text{---}
 \end{aligned} \tag{A.25}$$

For the first two circuits, Z commutes with $\Sigma_1^{W^\Sigma}$ so the $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$ blocks are equal and

$$\begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array} \begin{array}{c} L^\dagger \\ L^\dagger \\ L^\dagger \end{array} \begin{array}{c} \Sigma_1^{W^\Sigma} \\ \Sigma_1^{W^\Sigma} \\ \Sigma_1^{W^\Sigma} \end{array} \begin{array}{c} Z \\ Z \\ Z \end{array} \begin{array}{c} L \\ L \\ L \end{array} \begin{array}{c} X \\ X \\ X \end{array} \begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array} \text{---} = \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array} \begin{array}{c} L^\dagger \\ L^\dagger \\ L^\dagger \end{array} \begin{array}{c} C \\ C \\ C \end{array} \begin{array}{c} Z \\ Z \\ Z \end{array} \begin{array}{c} L \\ L \\ L \end{array} \begin{array}{c} X \\ X \\ X \end{array} \begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array} \text{---} = W_{00}^\Sigma = W_{11}^\Sigma$$

(A.26)

For the other two circuits, conjugating a diagonal matrix by a Pauli Z matrix does nothing, so we evaluate the $|0\rangle\langle 1|$ component to

$$\begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array} \begin{array}{c} L^\dagger \\ L^\dagger \\ L^\dagger \end{array} \begin{array}{c} \Sigma_2^{W^\Sigma} \\ \Sigma_2^{W^\Sigma} \\ \Sigma_2^{W^\Sigma} \end{array} \begin{array}{c} L \\ L \\ L \end{array} \begin{array}{c} X \\ X \\ X \end{array} \begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array} \text{---} = \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array} \begin{array}{c} \Sigma_2^U \\ \Sigma_2^U \\ \Sigma_2^U \end{array} \begin{array}{c} L^\dagger \\ L^\dagger \\ L^\dagger \end{array} \begin{array}{c} L \\ L \\ L \end{array} \begin{array}{c} X \\ X \\ X \end{array} \begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array} \text{---} = \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{c} \Sigma_2^U \\ \Sigma_2^U \\ \Sigma_2^U \end{array} \begin{array}{c} X \\ X \\ X \end{array} \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} = W_{01}^\Sigma.$$

(A.27)

The $|1\rangle\langle 0|$ component is the same with $-\Sigma_2^{W^\Sigma}$ so the $|1\rangle\langle 0|$ block evaluates to W_{10}^Σ .

Combine the calculations above to see that the CS Decomposition evaluates to

$$|0\rangle\langle 0| \otimes W_{00}^\Sigma + |1\rangle\langle 1| \otimes W_{11}^\Sigma + |0\rangle\langle 1| \otimes W_{01}^\Sigma + |1\rangle\langle 0| \otimes W_{10}^\Sigma = W^\Sigma \quad (\text{A.28})$$

which is what the lemma asserted. □