Microsoft

ROS: Resource-constrained Oracle Synthesis for Quantum Computers

Giulia Meuli, Mathias Soeken, Martin Roetteler, Giovanni De Micheli

@martinquantum
martinro@microsoft.com
microsoft.com/quantum

Motivation: oracle synthesis

Arithmetic:

Factoring: needs “constant” modular arithmetic

ECC dlogs: need generic modular arithmetic,
need to execute non-trivial straight-line programs

HHL: need sparse matrix access, need integer inverses; Newton type methods

Amplitude amplification:

Implementation of “oracles” / subroutines, e.g., for search, collision etc.

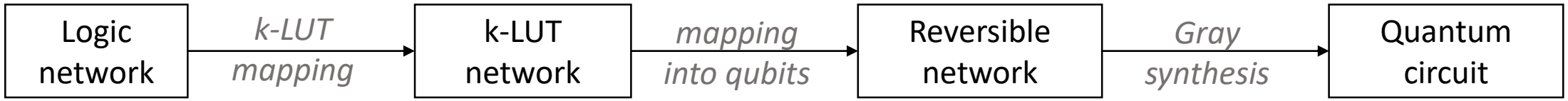
Implementation of walk operators on data structures

Quantum simulation:

Addressing/indexing functions for sparse matrices

Computing Hamiltonian terms on the fly

k-LUT based oracle synthesis

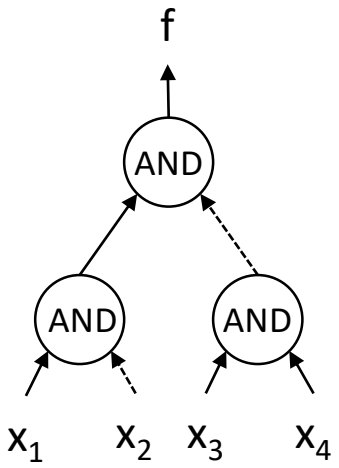


Logic Network

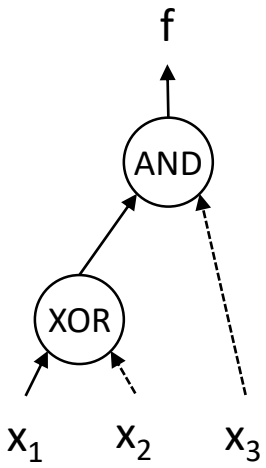
Hierarchical methods are based on logic networks rather than on less scalable logic representations, i.e. truth tables.

A logic network is a graph, in which each node represents a Boolean function and edges define data dependency.

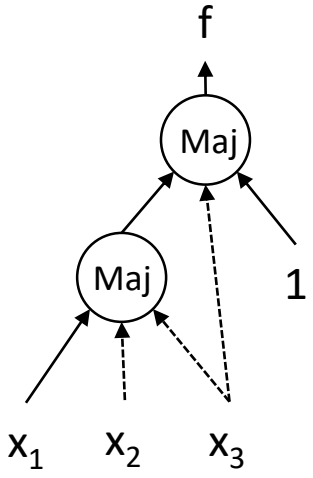
A network is characterized by the operations that it supports.



And-Inverter Graph (AIG)



Xor-And-inverter Graph (XAG)



Majority-Inverter Graph (MIG)

k-LUT mapping

Hierarchical methods use k-LUT mapping to decompose the original network.

A k-LUT is a subnetwork with maximum k inputs.

The decomposition is not unique.

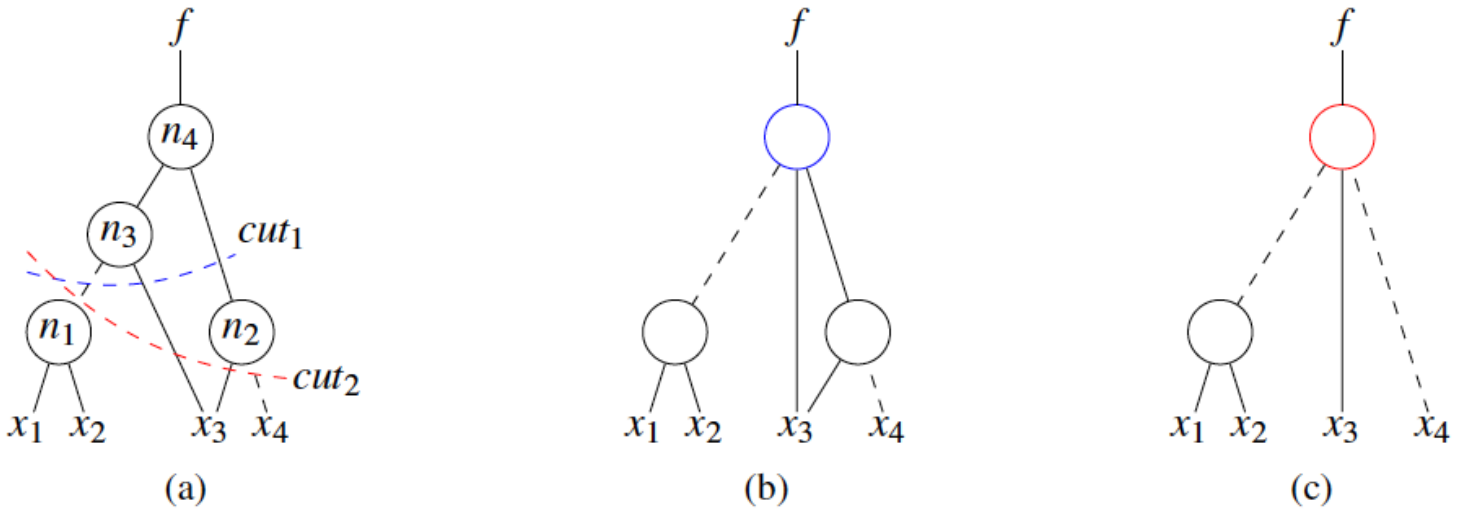
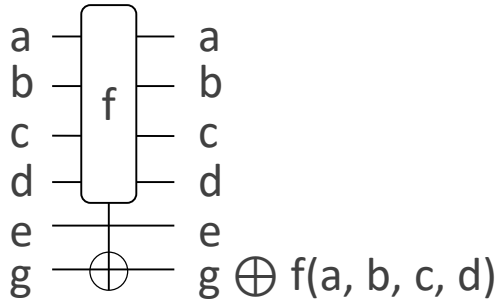


Figure 1: (a) an AIG graph performing the function $f = (\bar{x}_1 + \bar{x}_2)x_3\bar{x}_4$ with two possible 3-feasible cuts; (b) 3-LUT network generated by cut_1; (c) 3-LUT network generated by cut_2.

Mapping into qubits

This step exploits the one-to-one correspondence between a LUT node and a single-target gate.

Single target gate (STG)

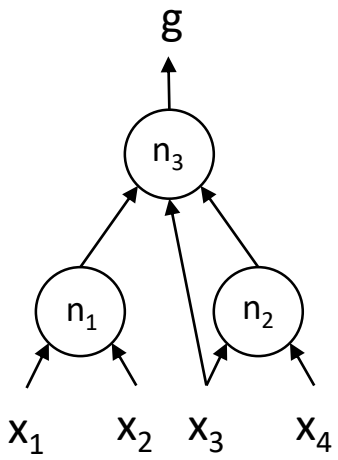


$T_c(\{x_1, \ldots, x_k\}, x_{k+1})$

$\{x_1, \ldots, x_k\}$ is a set of *control lines*

x_{k+1} is the *target line*

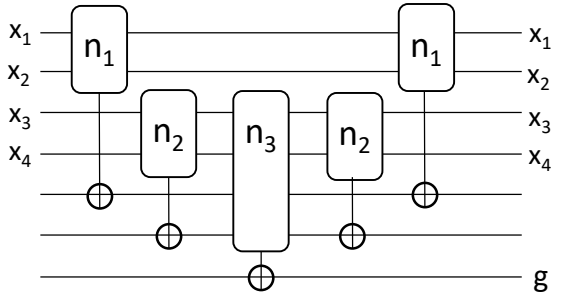
$f\colon B^k \to B$ is the *control function*



$n_1 = \mathrm{x}_1 \wedge \mathrm{x}_2$

$n_2 = x_3 \oplus \bar{x}_4$

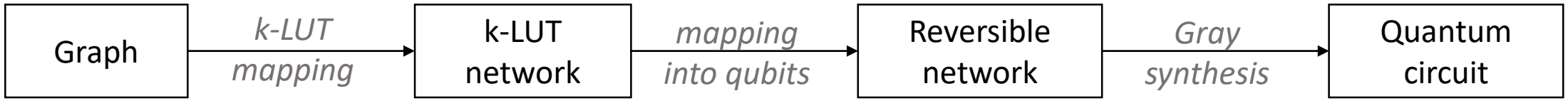
$n_3 = (n_1 \wedge n_2) \vee x_3$

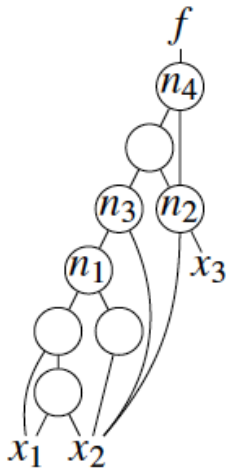


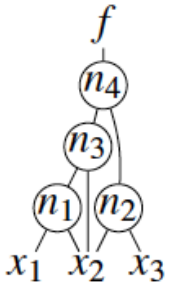
Each auxiliary qubit stores one intermediate result.

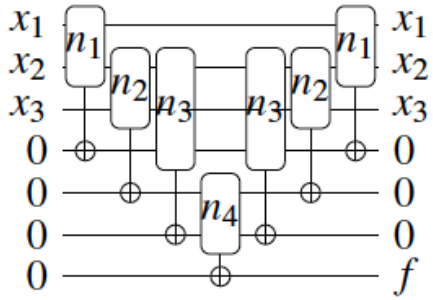
The order in which operations are computed affects the number of qubits and the number of gates.

k-LUT based oracle synthesis









During this step we define:

- Number of nodes in the k-LUT network
- Complexity of the function performed by each node

Gray synthesis: Phase polynomial

Computing a minimal parity network for a particular set of parity functions is shown to be equivalent to finding a CNOT-optimal circuit *for a particular phase polynomial*

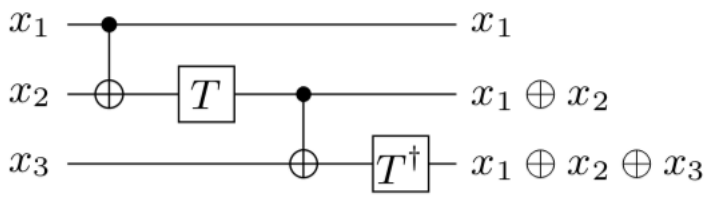
A {CNOT, R_Z} circuit has associated unitary:

$$U_C = \sum_{x \in F_2^n} e^{2\pi i f(x)} |A\boldsymbol{x}\rangle\langle \boldsymbol{x}|$$

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in F_2^n} \hat{f}(\boldsymbol{y})\ (x_1 y_1 \oplus x_2 y_2 \oplus \ldots \oplus x_n y_n)$$

The expression $f(x)$ is a weighted sum of parities in $\boldsymbol{x}$.

Example

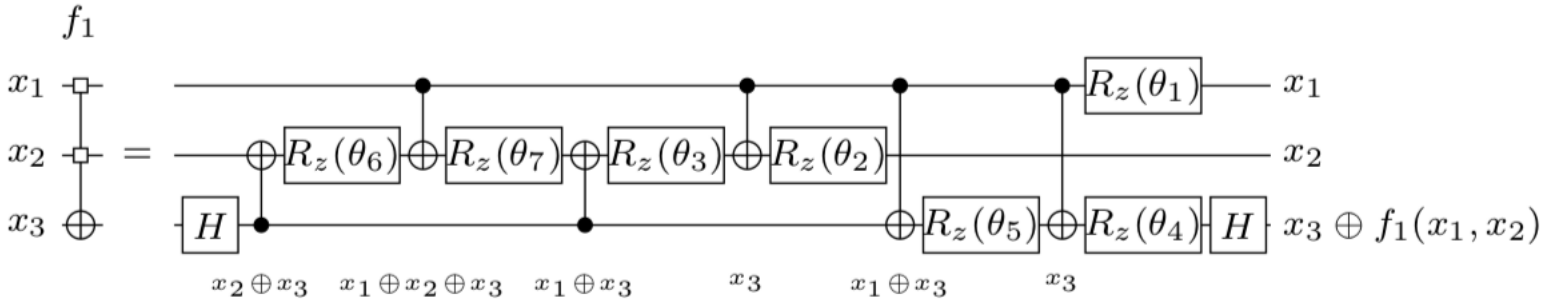


A: $|x_1\rangle|x_2\rangle|x_3\rangle \rightarrow |x_1\rangle|x_1 \oplus x_2\rangle|x_1 \oplus x_2 \oplus x_3\rangle$

$$f(x) = \frac{\pi}{4}(x_1 \oplus x_2) + \frac{7}{4}\pi(x_1 \oplus x_2 \oplus x_3)$$

M. Amy, P. Azimzadeh, and M. Mosca, "On the CNOT-complexity of CNOT-phase circuits," *arXiv:1712.01859*, 2017

Synthesis of single-target gates



When a STG performing $|x_1\rangle|x_2\rangle|x_3\rangle \rightarrow |x_1\rangle|x_2\rangle|x_1 \oplus f_1(x_1, x_2)\rangle$

is synthesized into a {CNOT, R_Z} circuit, the CNOTs are used to generate linear combinations of the inputs.

The rotation angles are proportional to the Rademacher-Walsh spectral coefficients of the function

$$\hat{g} = x_3 \wedge f_1(x_1, x_2)$$

$$\theta_i = \frac{\pi s_i}{2^n}$$

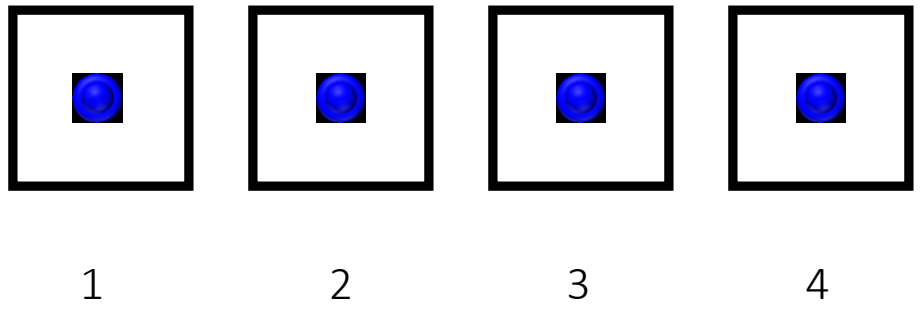
A function with many zeros as spectral coefficients will be translated into less gates.

Space-time tradeoffs: pebbling

Rules of the game: **[Bennett, SIAM J. Comp., 1989]**

- n boxes, labeled i = 1, …, n
- in each move, either add or remove a pebble
- a pebble can be added or removed in i=1 at any time
- a pebble can be added of removed in i>1 if and only if there is a pebble in i-1.

Example:



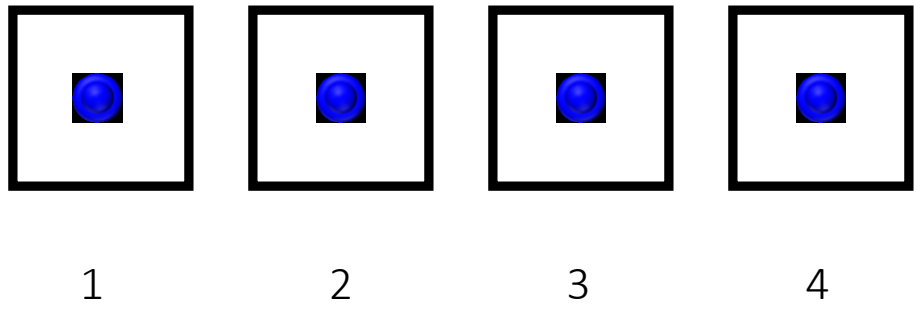
#	i
1	1
2	2
3	3
4	4
5	3
6	2
7	1

Space-time tradeoffs: pebbling

Imposing resource constraints:

- only a total of S pebbles are allowed
- corresponds to reversible algorithm with at most S auxiliary qubits

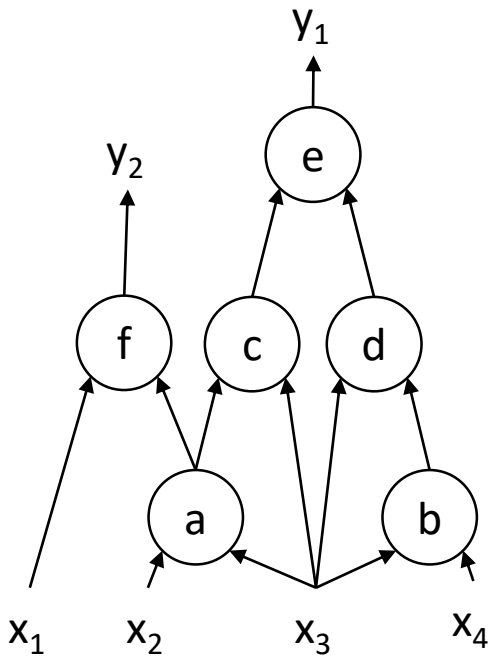
Example: (n=3, S=3)

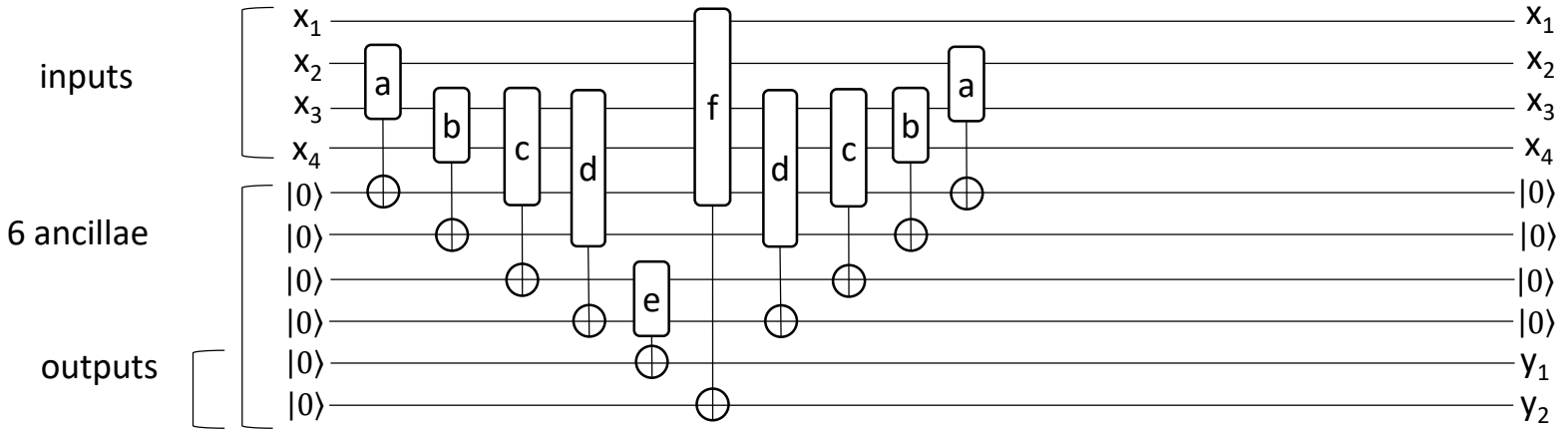


#	i
1	1
2	2
3	3
4	1
5	4
6	3
7	1
8	2
9	1

Mapping into qubits: pebbling strategy

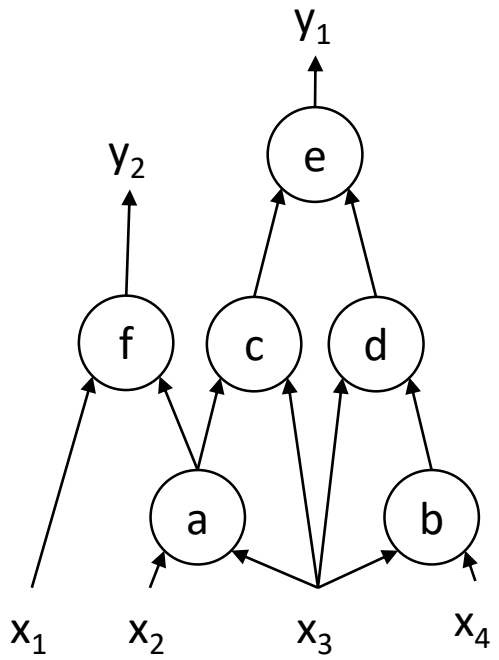
This step defines the number of auxiliary qubits.
It can be optimized by changing the order in which the graph is traversed.

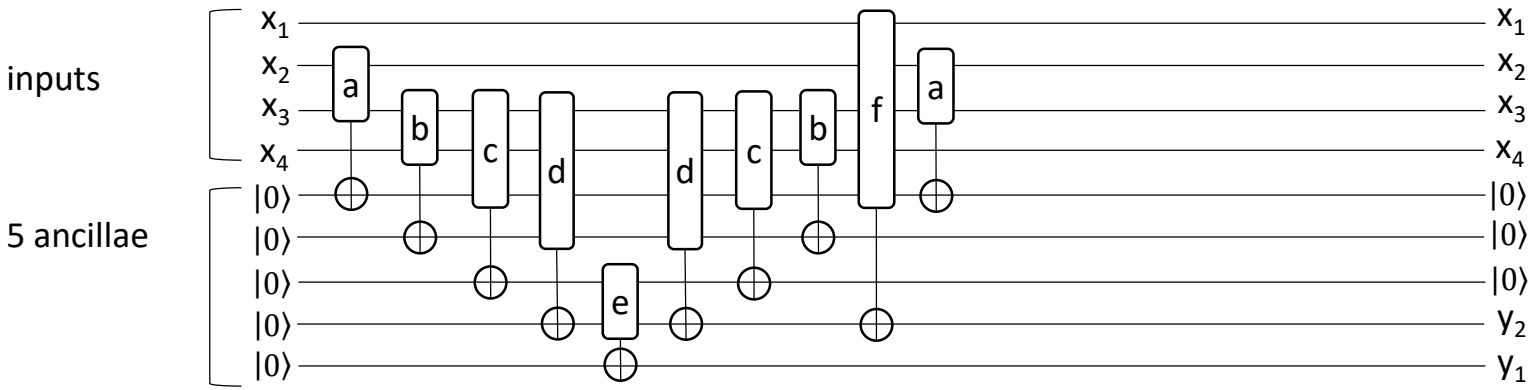




Mapping into qubits: pebbling strategy

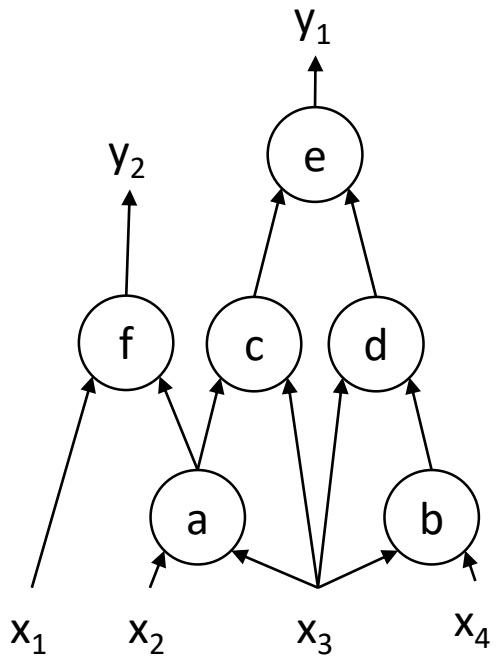
This step defines the number of auxiliary qubits.
It can be optimized by changing the order in which the graph is traversed.

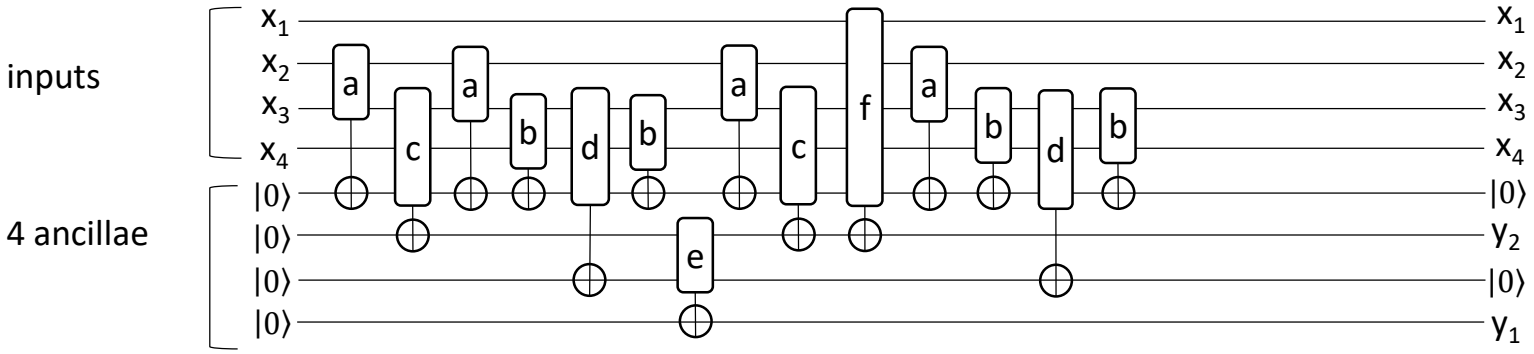




Mapping into qubits: pebbling strategy

This step defines the number of auxiliary qubits.
It can be optimized by changing the order in which the graph is traversed.





Results

	M/B		S/B		S/P_match_q		S/P_match_g		M/P	
	gates	qubits	gates	qubits	gates	qubits	gates	qubits	gates	qubits
addassoc4	1376	25	1029	34	1141	25	1371	22	1904	19
addassoc5	2987	36	1586	49	1798	36	1804	31	5365	24
addassoc6	2394	43	1445	58	1513	42	1729	35	8268	26
addassoc7	3243	51	1941	70	2201	50	2361	44	4383	36
addassoc8	3221	62	2018	79	2312	57	2430	49	4787	40
addassoc9	3603	70	2385	89	2453	67	2773	56	5569	42
addassoc10	4528	80	2835	97	3575	70	3549	58	6142	50
multassoc4	6682	34	2751	60	3057	34	3193	33	10834	19
multassoc5	10519	54	4811	104	5321	55	5321	55	16687	31
multassoc6	17653	93	7395	172	8565	96	8565	96	22933	53
multassoc7	25395	138	11099	240	15425	135	14607	128	37717	74
multassoc8	32443	181	13781	323	20713	179	22997	166	51757	94
multassoc9	37599	212	17881	394	34305	203	32489	200	66267	110
multassoc10	47795	289	22843	525	41825	281	41081	262	101627	143
multdistr4	4812	29	2368	54	3262	29	3694	25	5034	19
multdistr5	9011	54	4569	94	5441	54	5441	54	22717	25
multdistr6	13327	78	6092	143	7138	80	7138	80	15169	46
multdistr7	18268	110	8771	200	13849	109	13849	109	21746	63
multdistr8	26151	149	11888	276	17896	149	17520	143	39449	81
multdistr9	30427	184	14477	332	22917	182	22445	181	43819	99
multdistr10	37571	226	17808	414	29714	214	31570	219	55583	122
S/P vs M/B average results					-32.31	-1.86	-29.77	-8.38		

Thanks!

Microsoft Quantum Development

github.com/microsoft/quantu

Explore the documentation

docs.microsoft.com/quantum/

Further reading

microsoft.com/quantum