Reversible Logic Circuit Synthesis

Vivek V. Shende, Aditya K. Prasad,
Igor L. Markov and John P. Hayes
University of Michigan

Outline

- Motivation
  - Real-world Applications
  - Theoretical Advantages
  - Links to Quantum Computation
- Background
- Theoretical Results
- Synthesis of Optimal Circuits
- An Application to Quantum Computing

Real-world Applications

- Many inherently reversible applications
- Info. is re-coded, but none is lost or added
  - Digital signal processing
  - Cryptography
  - Communications
  - Computer graphics
  - Network congestion modeling

Theoretical Advantages

- Information conservation laws in physics
  - Thermodynamics ties irreversibility to dissipated heat: every lost bit causes an energy loss
  - C. Bennett, 1973, *IBM J. of R & D*
- Energy-lossless circuits ($Time \rightarrow \infty$)
  - must be information-lossless
  - have been built: S. Younis and T. Knight, 1994, *Workshop on Low Power Design*

Links to Quantum Computation

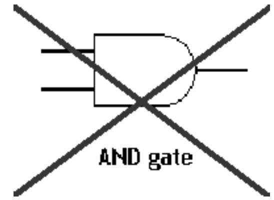
- Quantum operations are all reversible
  - M. Nielsen and I. Chuang, **Quantum Computation and Quantum Information**, *Cambridge Univ. Press* 2000
- Every (classical) reversible circuit may be implemented in quantum technology, with overhead
- "Pseudo-classical" subroutines of quantum algos
  - Can be implemented in classical reversible logic circuits
  - S. Betteli, L. Serafini, and T. Calarco, 2001, http://xxx.lanl.gov/abs/cs.PL/0103009

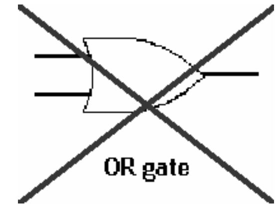
Outline

- Motivation
- Background
  - Reversibility
  - Permutations
  - Known Facts
- Theoretical Results
- Synthesis of Optimal Circuits
- An Application to Quantum Computing

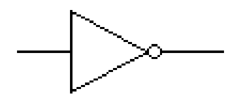
Reversibility in Logic Gates

- Definition: reversible logic gate
  - #input wires = #output wires
  - Permutes the set of input values

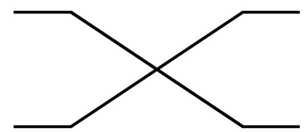




- Examples
  - Inverter
  - *2*-input, *2*-output SWAP (S) gate

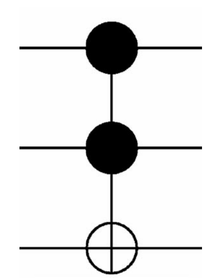


Inverter



SWAP gate

- *k*-CNOT gate
  - (*k+1*)-inputs and (*k+1*)-outputs
  - Values on the first *k* wires are unchanged
  - The last value is flipped if the first *k* were all 1



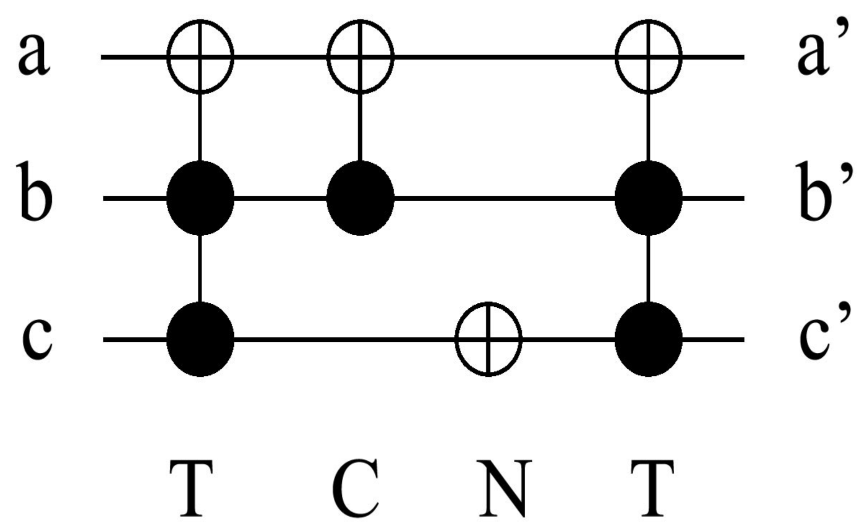
2-CNOT gate
(TOFFOLI)

Reversibility in Logic Circuits

- Definition:
  A combinational logic circuit is reversible iff
  - It contains only reversible gates
  - It has no fan-out
  - It is acyclic (as a directed multi-graph)
- Theorem:
  A reversible circuit must
  - Have as many input wires as output wires
  - Permute the set of input values

Example:
A Reversible Circuit and Truth Table

a	b	c	a'	b'	c'
0	0	0	0	0	1
0	0	1	0	0	0
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	1	1	0



Equivalent to a NOT gate on wire c

Circuit Equivalences

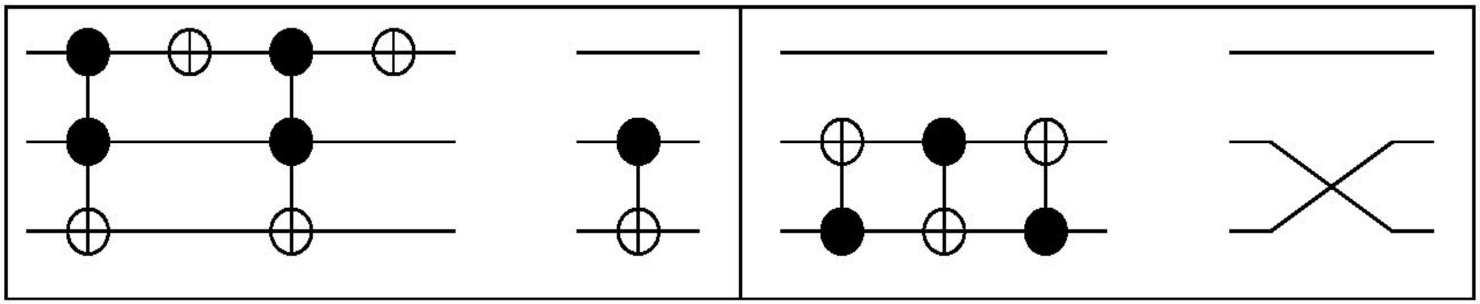


Figure 2: Two reversible circuit equivalences. $T(1,2;3) \cdot N(1) \cdot T(1,2;3) \cdot N(1) = C(2;3)$, **and** $C(3;2) \cdot C(2;3) \cdot C(3;2) = S(2,3)$

- Circuit equivalences: useful in synthesis
- More will be shown later

Reversible Circuits & Permutations

- A reversible gate (or circuit) with *n* inputs and *n* outputs has
  - 2^n possible input values
  - 2^n possible output values
- The function it computes on this set must, by definition, be a permutation
- The set of such permutations is called S_{2^n}

MichiganEngineering

Basic Facts About Permutations

- Permutations are multiplied by first applying one, then the other
  - example: $(1,2)\circ(2,3) = (1,3,2)$
- A transposition
  - permutes exactly two elements
  - does not change any others
- Every permutation can be written as a product of transpositions

Even Permutations

- Consider all possible decompositions of a permutation into transpositions
- Theorem: The parity of the number of transpositions is constant
- Definition: Even permutations are those for which the number of transpositions is even

Known Facts

- Fact 1: Consider a reversible circuit
  - $n+1$ inputs and $n+1$ outputs
  - Built from gates which have at most n inputs and n outputs
  - Must compute an even permutation
- Fact 2: A universal gate library
  - CNOT, NOT, and TOFFOLI ("CNT")
  - Temporary storage may be required

Temporary Storage

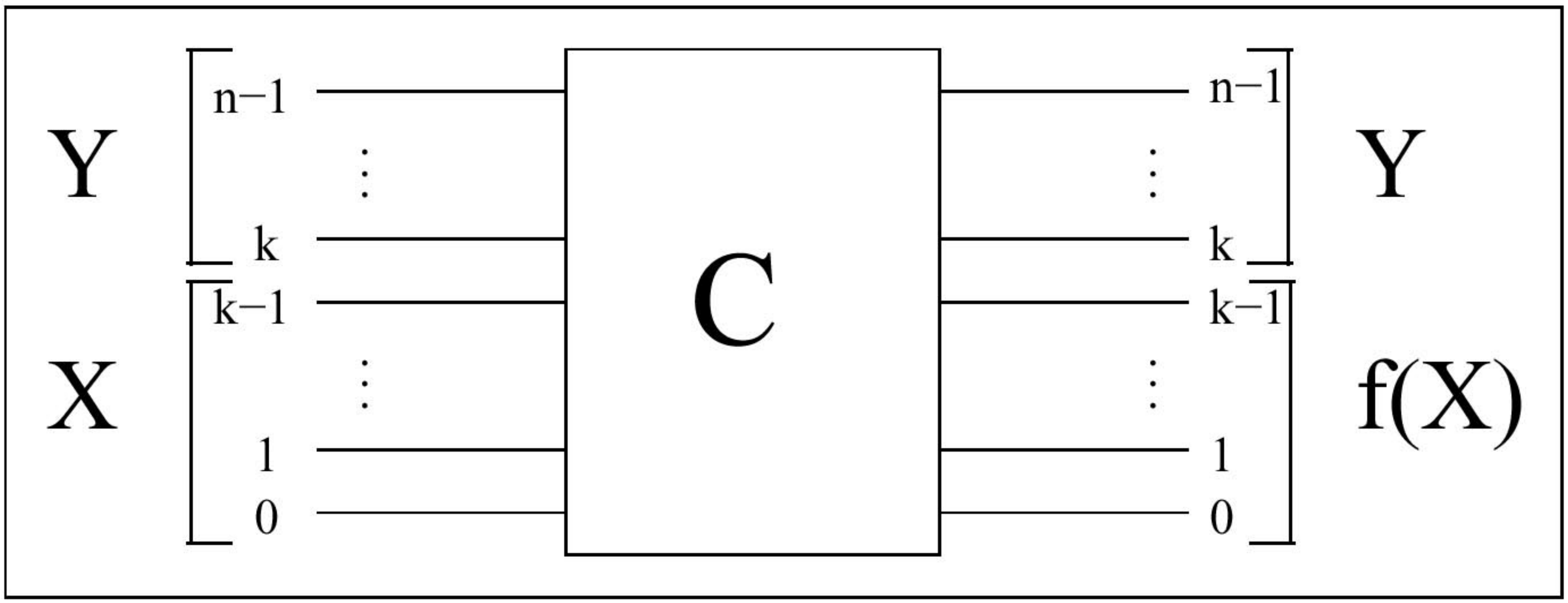


Figure 3: A reversible circuit with n-k wires of temp. storage.

Outline

- Motivation
- Background
- Theoretical Results
  - Zero-storage Circuits
  - Reversible De Morgan's Laws
- Synthesis of Optimal Circuits
- An Application to Quantum Computing

Minimizing Temporary Storage

- Consider CNT circuits
  - Theorem: even permutations computable by circuits without temporary storage
  - Theorem: odd permutations computable *with one line of temporary storage*
- Same holds for NT and CNTS circuits
- The proof is constructive and may be used as a synthesis heuristic

Outline of Proof

- Explicitly construct a circuit to compute an arbitrary pair of <u>disjoint</u> transpositions
  (A, B) (C, D) is okay; (A, B) (B, C) is not
- Pick an even permutation
- Decompose it into transpositions
  - Will have an even number of transpositions
- Pair these up, <u>guaranteeing disjointness</u>
- Apply above construction to each pair

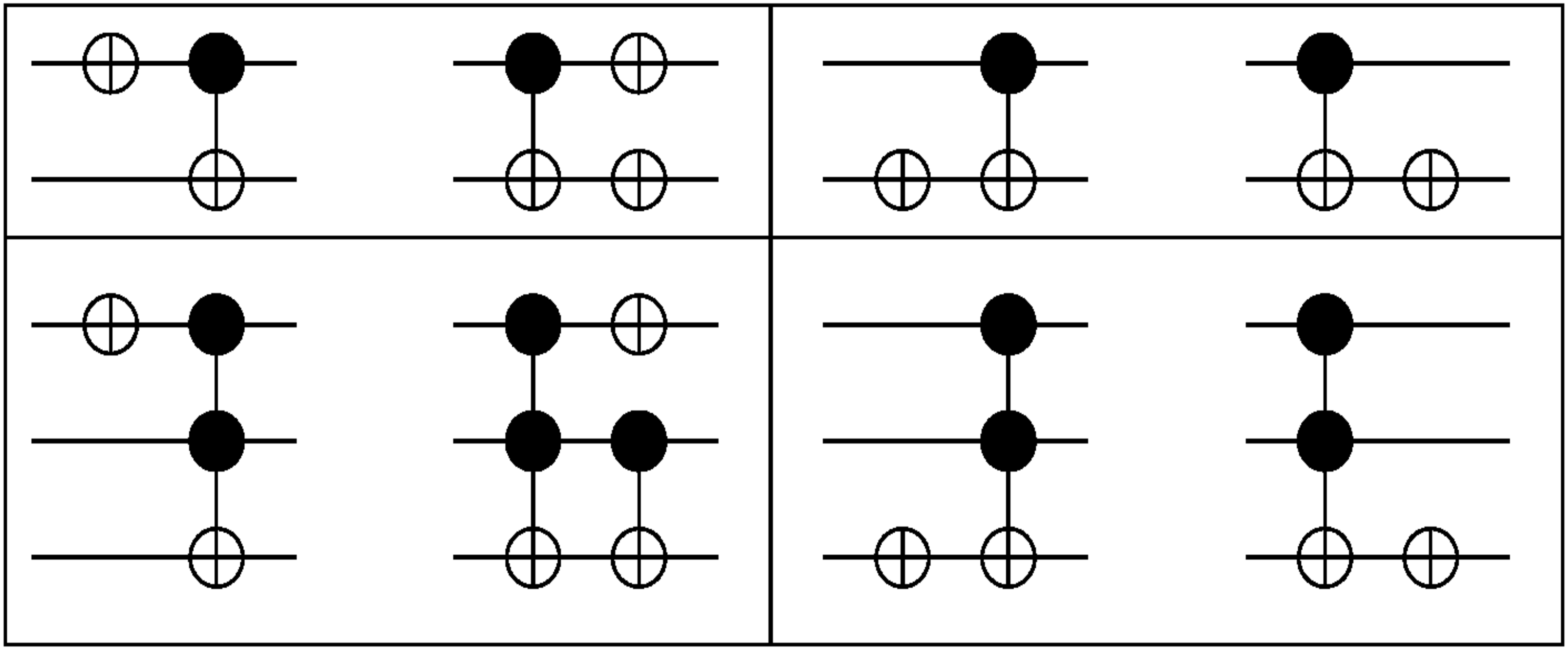
Reversible De Morgan's Laws (1)

- De Morgan's Laws
  - Apply to AND/OR/NOT circuits
  - Allow pushing all inverters to the inputs
- Reversible De Morgan's Laws
  - Applied to reversible CNT circuits
  - Allow pushing all inverters to the inputs
- Pictures follow

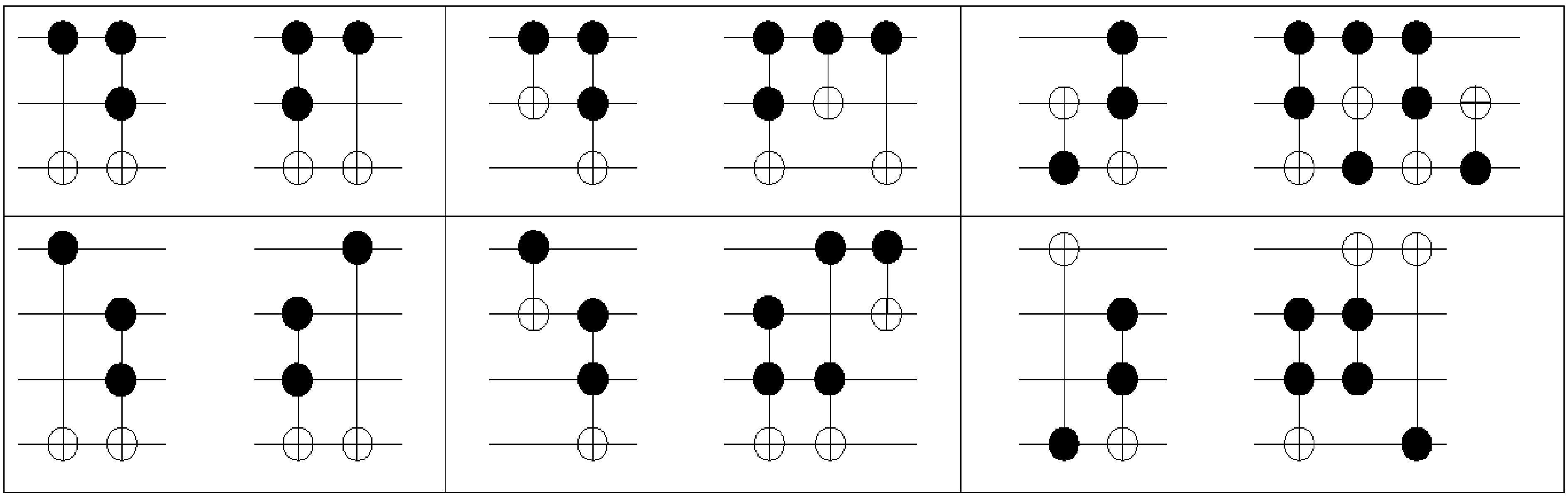
Reversible De Morgan's Laws (2)

- Similar rules exist for interchanging TOFFOLI and CNOT gates
- However, it is not always possible to push all CNOT gates to the inputs
- Oddly enough, all CNOT gates can be pushed to the "middle" of the circuit

Reversible De Morgan's Laws (3)



Reversible De Morgan's Laws (4)



Outline

- Motivation
- Background
- Theoretical Results
- Synthesis of Optimal Circuits
  - Optimality
  - IDA* Search Algorithm
  - Circuit Libraries
- An Application to Quantum Computing

Optimality

- The cost of a circuit is its <u>gate count</u>
  - Other cost functions can be considered
- <u>Definition</u>: optimal reversible circuit
  - no circuit with fewer gates computes the same permutation
- <u>Theorem</u>: a sub-circuit of an optimal circuit is optimal
  - Proof: otherwise, can improve the sub-circuit

IDA* Search

- Checks all possible circuits of cost 1, then all possible circuits of cost 2, etc…
- Avoids the memory blowup of BFS
- Still finds optimal solutions (unlike DFS)
- Checking circuits of cost less than *n*
  - Is much faster than processing cost-*n* circuits

Dynamic Prog + Circuit Libraries

- IDA* search requires a subroutine to check all circuits of cost *n*, for arbitrary *n*
  - Called iteratively for *1…n*
- Only need to check <u>locally optimal circuits</u>
- Build optimal circuit library bottom up by DP
  - Index optimal circuits by computed permutation
  - In practice use hash_map datastruct from STL

Synthesis Algorithm (1)

```
CIRCUIT synthesize(PERM)
{
if (PERM==IDENTITY) return EMPTY_CCT;
// otherwise, use IDA* to find a circuit.
for(DEPTH ← 1, DEPTH < MAX_DEPTH; DEPTH++)
    {
    CIRCUIT ← find_circ(DEPTH, PERM, EMPTY_CCT);
    if (CIRCUIT != NIL) return CIRCUIT;
    }
}
```

Synthesis Algorithm (2)

```
CIRCUIT find_circ(COST, PERM, CURR_CCT)
{
if (COST ≤ k)
   // if PERM can be computed by a circuit
   // with fewer at most k gates,
   // such a circuit must be in the library
   return CURR_CCT + LIB[DEPTH].find(PERM));
else
   // The goal circuit must have >k gates;
   // Try constructing it from k-gate circuits
   for each C in LIB[k]
   {
     // divide PERM by permutation computed by C
     PERM2 ← PERM * INVERSE(C.perm)
     // and try to synthesize the result
     TEMP_CCT ← find_circ(depth-k,PERM2));
     if (TEMP_CCT != NIL) return TEMP_CCT;
   }
}
```

MichiganEngineering

Empirical Circuit Synthesis

- Consider all reversible functions on 3 wires
(8! = 40,320 functions)
- For each gate library from
N, C, T, NC, CT, NT, CNT, CNTS
  - Is it universal?
  - How many functions can it synthesize?
  - What are largest optimal circuits?
  - How long does it take to synthesize circuits?

Optimal Circuit Sizes

Size	N	C	T	NC	CT	NT	CNT	CNTS
12	0	0	0	0	0	47	0	0
11	0	0	0	0	0	1690	0	0
10	0	0	0	0	0	8363	0	0
9	0	0	0	0	0	12237	0	0
8	0	0	0	0	6	9339	577	32
7	0	0	0	14	386	5097	10253	6817
6	0	2	0	215	1688	2262	17049	17531
5	0	24	0	474	1784	870	8921	11194
4	0	60	5	393	845	296	2780	3752
3	1	51	9	187	261	88	625	844
2	3	24	6	51	60	24	102	135
1	3	6	3	9	9	6	12	15
0	1	1	1	1	1	1	1	1
Total	**8**	**168**	**24**	**1344**	**5040**	**40320**	**40320**	**40320**
Time, s	**0**	**0**	**0**	**30**	**215**	**97**	**40**	**15**

MichiganEngineering

Outline

- Motivation
- Background
- Theoretical Results
- Synthesis of Optimal Circuits
- An Application to Quantum Computing
  - Grover's Search
  - Pseudo-classical Synthesis

Michigan Engineering

Grover's Search

- A quantum algorithm for associative search (input is not sorted)
  - Search criterion: a classical one-output function *f*
  - L. K. Grover, "A Framework For Fast Quantum Mechanical Algorithms", *STOC* 1998
  - M. Nielsen and I. Chuang, 2000
- Runs in time $O(\sqrt{N})$
  - any classical algorithm provably requires $\Omega(N)$ time
- Requires a subroutine (oracle) that
  - changes the phase (sign) of all basis states (bit-strings) that match the search criterion *f*

Pseudo-classical Synthesis

- We focus on circuits for Grover oracles
- To change the sign of a bit-string
  - Initialize a qubit to |0> - |1>
  - Compute the classical one-output function *f*
  - XOR the qubit with *f*
  - Whenever *f=1*, the sign (phase) will change
- Thus, the design of Grover search circuits for a given *f*
  - Is reduced to reversible synthesis
  - Can be solved optimally by our methods

ROM-based Circuits

- Desired circuits must alter phase of basis states
  - All bits except one must be restored to input values
- Previous work studied ROM-based circuits
  - Constraint: ROM qubits can never change
  - B. Travaglione et al., 2001, http://xxx.lanl.gov/abs/quant-ph/0109016
  - Theorems + heuristic synthesis algorithms
- Our work: synthesis of pseudo-classical circuits
  - 3 read-only “ROM” wires that can never change
  - 1 wire that can be changed during computation, but must be restored by end
  - 1 wire on which function is computed

Synthesis Algorithms Compared

- Heuristic synthesis of ROM-based circuits
  - Proposed by Travaglione et al, 2001
  - Based on EXOR-sum decomposition ("EXOR") (Our code uses Mishchenko's EXORCISM-4)
  - Imposed a restriction: at most one control bit per gate can be on a ROM bit
- Optimal synthesis (as described earlier)
  - with restriction from Travaglione ("Opt T")
  - without this restriction ("Opt")

Sizes of 3+2 ROM-circuits

Size	0	1	2	3	4	5	6	7	8	9	10	11	12
EXOR	1	4	6	4	4	12	18	12	6	12	19	16	10
OptT	1	4	6	4	4	12	21	24	29	33	44	46	22
Opt	1	7	21	35	36	28	28	36	35	21	7	1	0

Size	13	14	15	16	17	18	19	20	21	22	23	24	25	26
EXOR	8	10	16	19	12	6	12	18	12	4	4	6	4	1
OptT	5	1	0	0	0	0	0	0	0	0	0	0	0	0
Opt	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Discussion of Empirical Results

- The EXOR-SUM heuristic is sub-optimal
- All methods able to synthesize all 256 fns
  - "Opt T" able to synthesize as many as "Opt": B. Travaglione et al., 2001
- "Opt" results symmetrical about 5-6 gates
  - Function *x* requires one fewer gate than *256-x*
  - Explanation yet to be found
- "Exor" results symmetrical about 13 gates

Conclusions

- Classical reversible circuits
  as special-case quantum circuits
- Existence theorems
- Reversible De Morgan's laws
  - Future research on optimization heuristics
- Algorithm for synthesis of optimal circuits
  - Applicable to Grover's search

Thank You For Your Attention

MichiganEngineering