Quantum Logic Synthesis

(or, quantum computer compilation)

1

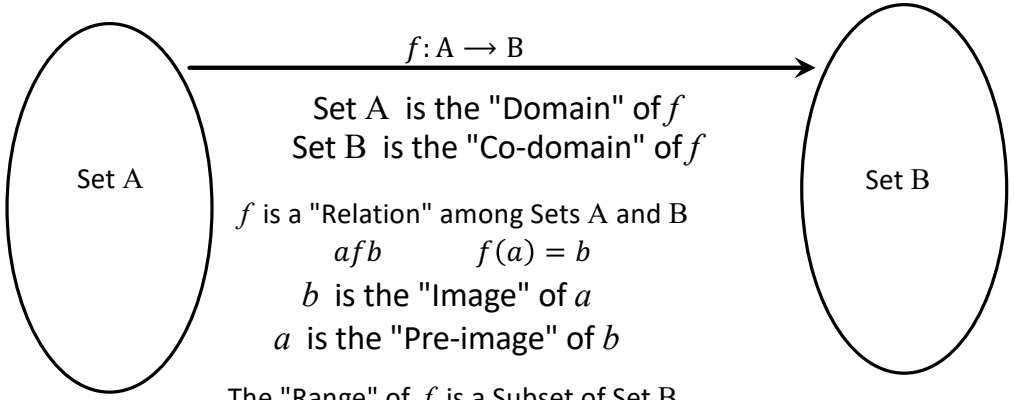
Synthesis Outline

- Irreversible into Reversible Switching Functions
  - Proof that Reversible Switching Functions can be Implemented as Quantum Circuits
  - The RTT Approach
- Reversible Switching Function into Reversible Operators
  - The MMD Approach
- Irreversible Specifications into Reversible Operators
  - The ESOP Approach
  - Structure of an Oracle
- Mapping Reversible Operators into Technology-Dependent Operators
  - Various Decompositions
- Representing Classical Data as State Preparation Circuits
  - QROM versus QRAM
  - Quantum Data Encoding
- Embedding Data within Quantum Programs/Circuits

2

Discrete Function Concepts

Function f Defines a Mapping of Elements in Set A to a Subset of Elements in Set B



Set A is the "Domain" of f
Set B is the "Co-domain" of f

f is a "Relation" among Sets A and B

$afb \qquad f(a) = b$

b is the "Image" of a

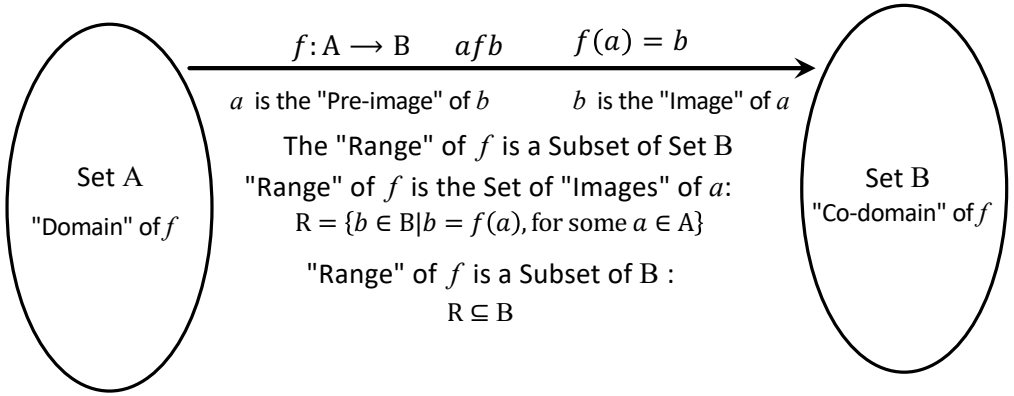
a is the "Pre-image" of b

The "Range" of f is a Subset of Set B

3

Discrete Function Concepts (cont.)

Function f Defines a Mapping of Elements in Set A to a Subset of Elements in Set B



The "Range" of f is a Subset of Set B

"Range" of f is the Set of "Images" of a:

$\mathrm{R} = \{b \in \mathrm{B} | b = f(a), \text{for some } a \in \mathrm{A}\}$

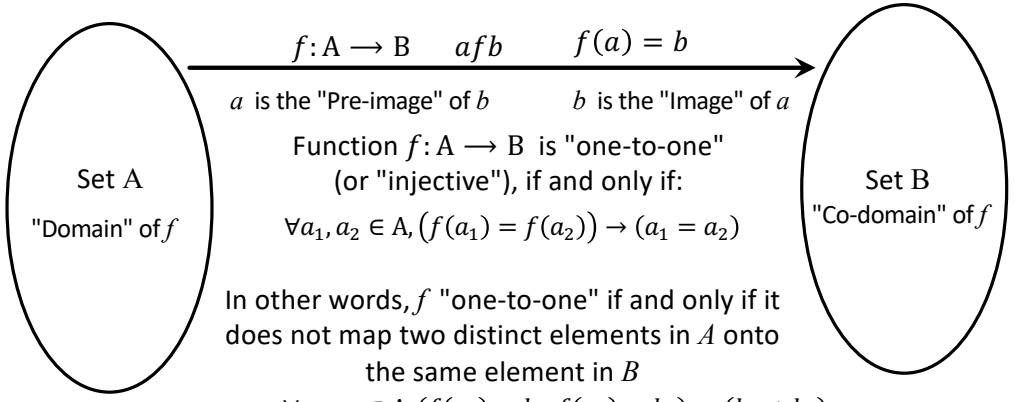
"Range" of f is a Subset of B :

$\mathrm{R} \subseteq \mathrm{B}$

4

Discrete Function Concepts (cont.)

Function f Defines a Mapping of Elements in Set A to a Subset of Elements in Set B



Function $f: \mathrm{A} \longrightarrow \mathrm{B}$ is "one-to-one" (or "injective"), if and only if:

$\forall a_1, a_2 \in \mathrm{A}, \big(f(a_1) = f(a_2)\big) \rightarrow (a_1 = a_2)$

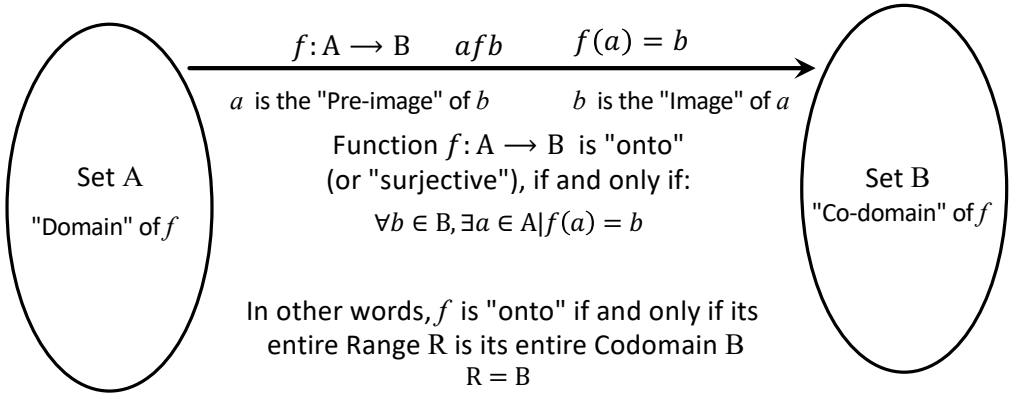
In other words, f "one-to-one" if and only if it does not map two distinct elements in A onto the same element in B

$\forall a_1, a_2 \in \mathrm{A}, (f(a_1) = b_1, f(a_2) = b_2) \rightarrow (b_1 \neq b_2)$

5

Discrete Function Concepts (cont.)

Function f Defines a Mapping of Elements in Set A to a Subset of Elements in Set B



Function $f: \mathrm{A} \longrightarrow \mathrm{B}$ is "onto" (or "surjective"), if and only if:

$\forall b \in \mathrm{B}, \exists a \in \mathrm{A} | f(a) = b$

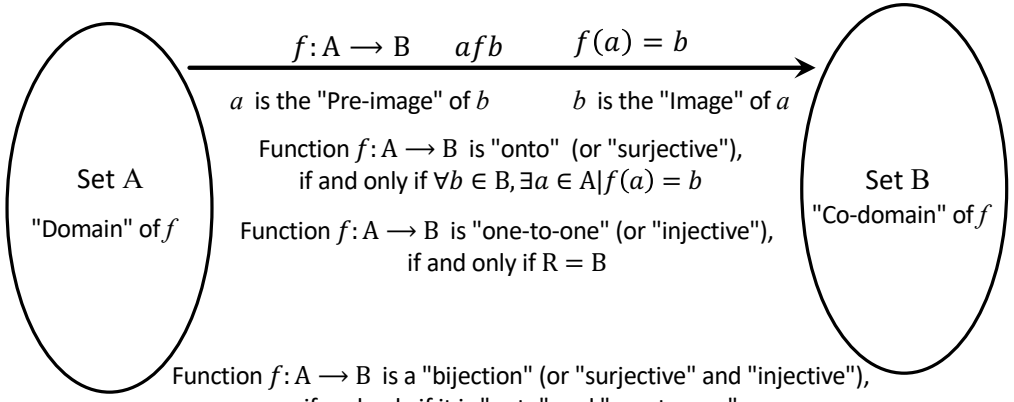
In other words, f is "onto" if and only if its entire Range R is its entire Codomain B

$\mathrm{R} = \mathrm{B}$

6

Discrete Function Concepts (cont.)

Function f Defines a Mapping of Elements
in Set A to a Subset of Elements in Set B



a is the "Pre-image" of b b is the "Image" of a

Function $f: \mathrm{A} \longrightarrow \mathrm{B}$ is "onto" (or "surjective"),
if and only if $\forall b \in \mathrm{B}, \exists a \in \mathrm{A} | f(a) = b$

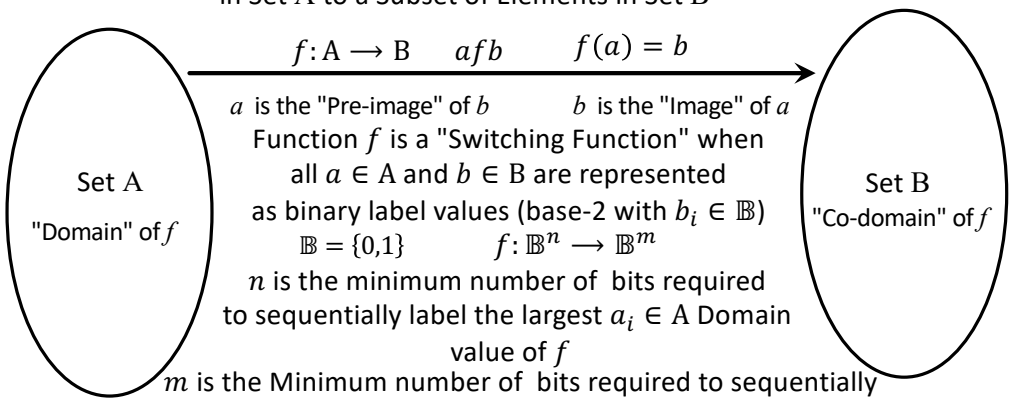
Function $f: \mathrm{A} \longrightarrow \mathrm{B}$ is "one-to-one" (or "injective"),
if and only if $\mathrm{R} = \mathrm{B}$

Function $f: \mathrm{A} \longrightarrow \mathrm{B}$ is a "bijection" (or "surjective" and "injective"),
if and only if it is "onto" and "one-to-one"

7

Discrete Switching Function Concepts

Function f Defines a Mapping of Elements
in Set A to a Subset of Elements in Set B



a is the "Pre-image" of b b is the "Image" of a

Function f is a "Switching Function" when
all $a \in \mathrm{A}$ and $b \in \mathrm{B}$ are represented
as binary label values (base-2 with $b_i \in \mathbb{B}$)
$\mathbb{B} = \{0,1\}$ $f: \mathbb{B}^n \longrightarrow \mathbb{B}^m$

n is the minimum number of bits required
to sequentially label the largest $a_i \in \mathrm{A}$ Domain
value of f

m is the Minimum number of bits required to sequentially
label the largest $b_i \in \mathrm{B}$ range value of f

8

Discrete Switching Function Concepts (cont.)

Function f Defines a Mapping of Elements
in Set $\mathrm{A} = \mathbb{B}^n$ to a Subset of Elements in Set $\mathrm{B} = \mathbb{B}^m$

Lemma 1: If $f: \mathbb{B}^n \longrightarrow \mathbb{B}^m$ is a bijection then $n = m$.

Proof: By definition, f must be one-to-one if it is bijective, thus the cardinality of the codomain of f, $|\mathbb{B}^m|$, must be greater than or equal to the cardinality of the domain of f, $|\mathbb{B}^n|$, that is, $|\mathbb{B}^m| \geq |\mathbb{B}^n|$ to ensure the one-to-one correspondence of each unique pair of pre-image and image values. Furthermore, since f is a switching function, $|\mathbb{B}^n| = 2^n$ and $|\mathbb{B}^m| = 2^m$. Additionally, since f is a bijection, it must have the onto property and its range R is equal to its codomain; therefore, the cardinality of the range is also $|\mathrm{R}| = |\mathbb{B}^m| = 2^m$. Given that m is the minimum number of bits for the largest-valued binary string that sequentially labels the image values of f and n is the minimum number of bits for the largest-valued binary string that sequentially labels the preimage values of f, it follows that $|\mathbb{B}^n| = |\mathbb{B}^m| = |\mathrm{R}|$. Thus, $2^m = 2^n$ proving the lemma that $n = m$.

9

Discrete Switching Function Concepts (cont.)

Function f Defines a Mapping of Elements
in Set $\mathrm{A} = \mathbb{B}^n$ to a Subset of Elements in Set $\mathrm{B} = \mathbb{B}^m$

Lemma 2: If $f: \mathbb{B}^n \longrightarrow \mathbb{B}^m$ is a bijection then it is of the form of permutation function wherein each preimage of f is mapped to a corresponding image value in accordance with a permutation relation.

Proof: From Lemma 1, the cardinality of the range and domain of f are equivalent and equal to 2^n. Since f is a switching function, the domain is comprised of a set of n-bit binary strings that are equivalent to the set of n-bit binary strings comprising the range. Furthermore, since f is a bijection and a switching function, each preimage is in the form of an n-bit binary string and each image is accordingly in the form of an n-bit binary string (*i.e.*, $m = n$), wherein each pair of preimage and corresponding image values are unique. Therefore, f is a permutation relation.

10

Discrete Switching Function Concepts (cont.)

Function f Defines a Mapping of Elements
in Set $\mathrm{A} = \mathbb{B}^n$ to a Subset of Elements in Set $\mathrm{B} = \mathbb{B}^m$

Lemma 3: If $f: \mathbb{B}^n \longrightarrow \mathbb{B}^m$ is a bijection in the form of permutation, then it can be represented by a $2^n \times 2^n$ permutation matrix, $\mathbf{U}$, that maps each pre-image to its corresponding image.

Proof: We form a 2^n-dimensional column vector where the components are labeled with n-bit strings beginning with the topmost component represented by all zeros and each subsequent lower value represented by increasing n-bit values with the lowermost component labeled by the all-ones n-bit label. In this way, each component represents a unique value in the domain of f. The function f can thus map each pre-image encoded as a 2^n-dimensional column vector into its corresponding image value that is also represented as a 2^n-dimensional column vector via a linear transformation defined by $\mathbf{U}$. Because f is a bijection and thus a permutation mapping from Lemma 2, $\mathbf{U}$ is accordingly a permutation matrix.

11

Discrete Switching Function Concepts (cont.)

Function f Defines a Mapping of Elements
in Set $\mathrm{A} = \mathbb{B}^n$ to a Subset of Elements in Set $\mathrm{B} = \mathbb{B}^m$

Definition 1: A reversible logic function, f, is a bijection wherein its corresponding inverse function, f^{-1}, is also a bijection.

Lemma 4: If $f: \mathbb{B}^n \longrightarrow \mathbb{B}^m$ is a bijection, then it is a reversible logic function.

Proof: From the result of Lemma 3, f can be represented by a permutation matrix $\mathbf{U}$. It is known that permutation matrices such as $\mathbf{U}$ have unique inverses that are also permutation matrices, thus f is a reversible logic function.

Theorem: If $f: \mathbb{B}^n \longrightarrow \mathbb{B}^m$ is a bijection, then it can be represented as a cascade of reversible logic elements suitable for implementation as a quantum circuit.

Proof: From Lemma 4, f is a reversible function represented by a permutation matrix $\mathbf{U}$. It is known that $\mathbf{U}\mathbf{U}^{-1} = \mathbf{I}$, and that $|\mathbf{U}| = 1$, thus $\mathbf{U}$ is unitary and implementable as a quantum circuit.

12

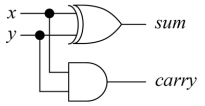
Embedding an Irreversible into Reversible Function

- **GOAL**: Convert Irreversible Switching Function to Reversible Switching Function with Minimal Number of Ancilla and Garbage Bits
- **MOTIVATION**: Functions are Often Embedded in Quantum Algorithms/Circuits as Part of an Oracle – Must be Reversible to Implement
  - QFT: Transformation Happened to be Unitary
  - Shor's Factoring: Modular Power Function Happened to be Unitary
  - What if the Function is Not Unitary? (*eg*. Grover's Oracle and Others?)
- One Approach is to Embed Irreversible Function into a Bijective Function
  - Add Ancilla/Garbage to Ensure Same Number of Domain/Range Bits
    - Necessary (but not sufficient) Condition for Function to be One-to-One
  - Desirable to Assign Ancilla Bits to Zero to adhere to gate-model QC property (initialization)
  - Must Assign Garbage Bits to Values to Ensure Function Comprised of Unique Mappings of Domain to Codomain – ensures that a bijective function results
    - Sufficient Condition for Function to be One-to-One and Onto (a Bijection)

13

Example of Embedding an Irreversible Function

- Consider a classical half-adder circuit



$sum = x \oplus y$

$carry = xy$

x	y	sum	$carry$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Repeated Image values; non-bijective

Irreversible Table

- Need to add at least one Garbage, which in turn requires an Ancilla
- Ancillas should be set to Zero to Ease Initialization

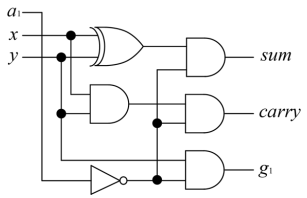
a_1	x	y	sum	$carry$	g_1
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	0
0	1	1	0	1	1

Reversible Table

$sum = a_1(x \oplus y)$

$carry = a_1 xy$

$g_1 = a_1 y$



We will address how to synthesize into a quantum circuit later

14

Reversible Truth Table (RTT) Method*

- Automated Method for Embedding Irreversible Switching Functions into Reversible Switching Functions
- Previous Example Done Manually, but this is Difficult for Larger Functions
- RTT is a Method that Gives Minimal Additional Qubits - Provably
  1) Mark Duplicate Output m-tuples
  2) Table Expansion
  3) One-to-one Mapping Assignments
- Example of RTT uses Truth Tables
  - More Efficient Switching Function Representations Reduces Complexity
    - Decision Diagrams (BDD, ADD, MDD, QMDD), Positional Cube Notation (PCN) lists/set. **`.pla`** Files
  - Representing Switching Functions are Worst-case Exponential Complexity, $O(2^n)$
  - Minimize Extra Ancilla/Garbage Qubits
    - Set Ancilla Qubits to Zero to Facilitate Quantum Algorithm Initialization

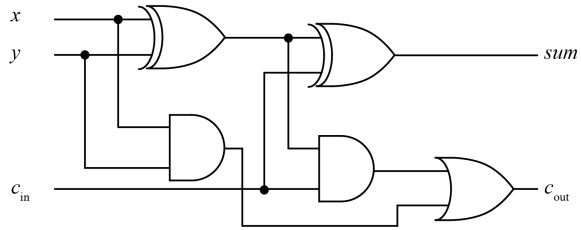
*E. Gabrielsen and M.A. Thornton, "Minimizing Ancilla and Garbage Qubits in Reversible Functions," in proc. Southwest Quantum Information and Technology, 20th Annual SQuInT Workshop, February 22-24, 2018.

15

RTT* – Example: Irreversible Full Adder

- Example: Irreducible Form of Full Adder

x	y	c_{in}	sum	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



*E. Gabrielsen and M.A. Thornton, "Minimizing Ancilla and Garbage Qubits in Reversible Functions," in proc. Southwest Quantum Information and Technology, 20th Annual SQuInT Workshop, February 22-24, 2018.

16

RTT* – Marking Duplicate Entries

- Example: Irreducible Form of Full Adder

```
def mark_duplicate_tuples(output):
    seen = set()
    duplicates = {}
    index = 0
    for row in output:
        t = tuple(row)
        if t in seen:
            if t in duplicates:
                duplicates[t].append(index)
            else:
                duplicates[t] = [index]
        seen.add(t)
        index += 1
```

x	y	c_{in}	sum	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Irreversible Table

The number of identical f_i ***m***-tuples are counted and marked in a two-dimensional array by iterating through each specified ***m-tuple*** in the f_i specification. The final dictionary represents a summary of the overall number of unique "keys", the number of key duplicates, and the correspondence of each ***m-tuple*** with the corresponding ***n-tuple*** valuation.

*E. Gabrielsen and M.A. Thornton, "Minimizing Ancilla and Garbage Qubits in Reversible Functions," in proc. Southwest Quantum Information and Technology, 20th Annual SQuInT Workshop, February 22-24, 2018.

17

RTT* – Table Expansion

- Example: Irreducible Form of Full Adder

```
def expand_table(input, output, dict={}):
    max_key = max(dict, key= lambda x: len(set(dict[x])))

    num_garbage = len("{0:b}".format(len(dict[max_key])))
    num_antica = (num_garbage + output.shape[1]) - input.shape[1]

    new_input = np.hstack((input, np.zeros((input.shape[0],
        num_antica), dtype=input.dtype)))

    new_output = np.hstack((output, np.zeros((output.shape[0],
        num_garbage)), dtype=input.dtype)))
```

a_0	x	y	c_{in}	sum	c_{out}	g_0	g_1
0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	1	0	1	0	0	0
0	0	1	1	0	1	0	0
0	1	0	0	1	0	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	1	1	0	0

Expanded Table

The minimum number of ancilla and garbage bits are determined. RTT expands table to hold placeholder values in order to ensure the second condition for function reversibility: for every input ***x*** there must exist some output ***y***

*E. Gabrielsen and M.A. Thornton, "Minimizing Ancilla and Garbage Qubits in Reversible Functions," in proc. Southwest Quantum Information and Technology, 20th Annual SQuInT Workshop, February 22-24, 2018.

18

RTT* – One-to-one Mapping Assignments

- Example: Irreducible Form of Full Adder

```
def one_to_one_mapping(input, output, num_antica, num_garbage, duplicates={}):
    buffer = output.shape[1] - num_garbage
    get_bin = lambda x, n: format(x, 'b').zfill(n)
    for key in duplicates:
        number = 1
        for item in duplicates[key]:
            garbage = get_bin(number, num_garbage)
            for col in range(num_garbage):
                output[item][col+buffer] = int(garbage[col])
            number = number + 1
    return (input, output)
```

Ensures the one-to-one rule holds for functional reversibility. Each ***n***-tuple and corresponding ***m***-tuple are iterated over to ensure unique rows. If a row is duplicated, the garbage or antica bit group is increased by 1 in binary.

a_0	x	y	c_{in}	sum	c_{out}	g_0	g_1
0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	1	0	1	0	0	1
0	0	1	1	0	1	0	0
0	1	0	0	1	0	1	0
0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	0
0	1	1	1	1	1	0	0

Reversible Table

*E. Gabrielsen and M.A. Thornton, "Minimizing Ancilla and Garbage Qubits in Reversible Functions," in proc. Southwest Quantum Information and Technology, 20th Annual SQuInT Workshop, February 22-24, 2018.

19

RTT* – Experimental Results

- From **`.pla`** Benchmarks Set

The RTT method was validated for functionality and effectiveness in an environment where benchmark irreversible functions were created and mapped to reversible form. We then compared our results to those from the RevLib collection.

Benchmark	Build Time (sec)	Run Time (sec)	Input	Output	Ancilla	Garbage	RevLib Ancilla	RevLib Garbage
Alu1	0.261	0.0361	12	8	6	10	8	12
Rd84_	0.139	0.00158	8	4	3	7	4	8
Addm4	0.222	0.00286	9	8	4	5		
Log8mod	0.0232	0.00147	8	5	2	5		
B12	43.36	0.398	15	9	7	13		
Cm162a	2.52	0.147	14	5	5	14		
Con1	0.00376	0.00136	7	2	1	6	2	7
Dc2	0.0269	0.00148	8	7	6	7	7	8

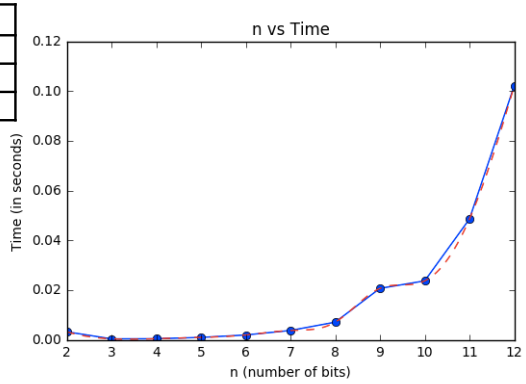
*E. Gabrielsen and M.A. Thornton, "Minimizing Ancilla and Garbage Qubits in Reversible Functions," in proc. Southwest Quantum Information and Technology, 20th Annual SQuInT Workshop, February 22-24, 2018.

20

RTT* – Complexity Analysis

Overall Performance of RTT is $O(2^n)$ due to the data structure being in matrix form. Future research is needed to reduce complexity closer to the theoretical minimum $O(N_{dup}N_{garb})$

Routine	Complexity
Marking Duplicate ***m***-tuples	$O(2^n)$
Table Expansion	$O(N_{dup}log(N_{dup}))$
One to One Mapping	$O(N_{dup}N_{garb})$



*E. Gabrielsen and M.A. Thornton, "Minimizing Ancilla and Garbage Qubits in Reversible Functions," in proc. Southwest Quantum Information and Technology, 20th Annual SQuInT Workshop, February 22-24, 2018.

21

RTT* - References

[1] IBM Corporation, "QISKit: Compiling and Running a Quantum Pro- gram", https://github.com/QISKit/qiskit-tutorial/blob/master/1 introduction/compiling_and_running.ipynb [2] K.N. Smith and M.A. Thornton, "MUSTANG-Q: A Technology Dependent Quantum Logic Synthesis and Compilation Tool," *Tech. Report*, Southern Methodist University, November 10, 2017.

[3] D. Deutsch, "Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer," in proc. *Royal Soc. of London A: Mathematical, Physical and Engineering Sciences,* vol. 400, no. 1818, pp. 97-117, 1985.

[4] A.W. Cross, L.S. Bishop, J.A. Smolin, and J.M. Gambetta, "Open Quantum Assembly Language," January 10, 2017, https://arxiv.org/pdf/1707.03429.pdf (accessed November 12, 2017).

[5] K. Fazel, M.A. Thornton, and J.E. Rice, "ESOP-based Toffoli Gate Cascade Generation," in proc. *IEEE Pac. Rim Conf. on Communications, Computers and Signal Processing,* pp. 206-209, 2007.

[6] S. Hassoun and T. Sasao, Logic Synthesis and Verification, Kluwer Academic Publishers, ISBN 978-1-4613-5253-2, 2002.

[7] R.E. Bryant, "Graph-based Algorithms for Boolean Function Manipulation," *IEEE Trans. Comp.,* 100(8), pp. 677-691, 1986.

[8] Vandana Maheshwari, *Development of SyReC Based Expandable Reversible Logic Circuits* Rajasthan, India: Rajasthan Technical University, 2014.

[9] Gorden E. Moore, *The Future of Integrated Electronics* Fairchild Semi- conductor internal publication, 1964.

[10] R. Landauer *Irreversibility and heat generation in the computing process*, IBM J. Res. Dev., vol. 5, p. 183, 1961.

[11] R. Drechsler and R. Wille, *From Truth Tables to Programming Languages: Progress in the Design of Reversible Circuits*, Bremen, Germany: University of Bremen, 2011.

[12] David Y. Feinstein and Mitchell A. Thornton, *On the Guidance of Reversible Logic Synthesis by Dynamic Variable Reordering*, Dallas, Tx: Southern Methodist University.

*E. Gabrielsen and M.A. Thornton, "Minimizing Ancilla and Garbage Qubits in Reversible Functions," in proc. Southwest Quantum Information and Technology, 20th Annual SQuInT Workshop, February 22-24, 2018.

22

Miller-Maslov-Dueck (MMD*) Method
(aka the "Table Based Method," TBS)

- Input is Reversible Switching Function Truth Table
- Output is Reversible Quantum Circuit using Operators:
  - Pauli-X
  - Controlled-Pauli-X, CNOT, Feynman Gate
  - Toffoli Gate, Controlled-Controlled-Pauli-X
  - Generalized Toffoli Gate (Controlled-Pauli-X with 3 or more control points)
- Requires Input Specification to be A Reversible (Bijective) Switching Truth Table
- Exponential Representation Complexity
- Later Versions of MMD/TBS used BDDs for Reduced Average-case Complexity but still has Exponential Worst-case Complexity
- Theory is to Reduce Transform "Residual" Function into an Identity

*D.M. Miller, D. Maslov and G.W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," in proc. IEEE/ACM Design Automation Conference (DAC), June 2-6, 2003.

23

Miller-Maslov-Dueck (MMD*) Method

- Example: Reversible Switching Function Truth Table

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

MMD: Output to Input Cascade Stage

1) Apply Pauli-X to Transform f(00...0)=00...0
2) Process each Output Word in Order Corresponding to Least Significant Input Word
3) And in Order of LSb to MSb in Output Word
4) Goal is to Make Output Word Equal Input Word
5) Use "Fixed" Bits to Transform "Unfixed" Bits in Output Word with {X, T, Gen. T} Gates from Right to Left
6) Repeat Until Output Words Match Input Words

*D.M. Miller, D. Maslov and G.W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," in proc. IEEE/ACM Design Automation Conference (DAC), June 2-6, 2003.

24

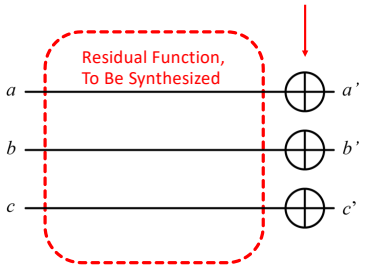
Miller-Maslov-Dueck (MMD*) Method

Step 1: Transform $f(00\ldots00)$ to $00\ldots0$

Residual Function

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

pre-image to image is an identity



Force the output to be all zero for the 000 input term

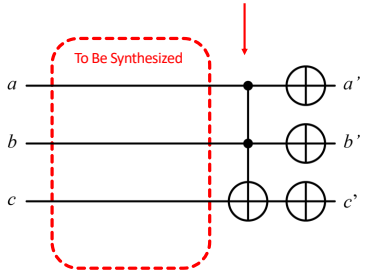
*D.M. Miller, D. Maslov and G.W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," in proc. IEEE/ACM Design Automation Conference (DAC), June 2-6, 2003.

25

Miller-Maslov-Dueck (MMD*) Method

Step 2: Change c^0 for Input Term 001 since a^0b^0=11

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



Choose LSb of Input Term 001 to Change

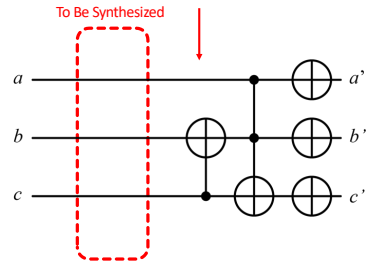
*D.M. Miller, D. Maslov and G.W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," in proc. IEEE/ACM Design Automation Conference (DAC), June 2-6, 2003.

26

Miller-Maslov-Dueck (MMD*) Method

Step 3: Change b^2 for Input Term 001 since c^1=1

a	b	c	a'	b'	c'	a^0	b^0	c^0	a^1	b^1	c^1	a^2	b^2	c^2
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	1	1	0	1	1	1	1	0	1
0	1	0	1	0	0	0	1	1	0	1	1	0	0	1
0	1	1	0	1	1	1	0	0	1	0	0	1	0	0
1	0	0	0	0	0	1	1	1	1	1	0	1	1	0
1	0	1	0	1	0	1	0	1	1	0	1	1	1	1
1	1	0	1	1	0	0	0	1	0	0	1	0	1	1
1	1	1	1	0	1	0	1	0	0	1	0	0	1	0



Must Synthesize b^2 First, then a^2

Choose "next" LSb of Input Term 001 to Change

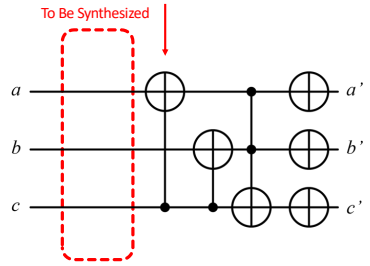
*D.M. Miller, D. Maslov and G.W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," in proc. IEEE/ACM Design Automation Conference (DAC), June 2-6, 2003.

27

Miller-Maslov-Dueck (MMD*) Method

Step 4: Change a^3 for Input Term 001 since c^2=1

a	b	c	a'	b'	c'		a^2	b^2	c^2	a^3	b^3	c^3	
0	0	0	1	1	1		0	0	0	0	0	0	← Identity
0	0	1	0	0	1		1	0	1	0	0	1	← Identity
0	1	0	1	0	0		0	0	1	1	0	1	
0	1	1	0	1	1		1	0	0	1	0	0	
1	0	0	0	0	0		1	1	0	1	1	0	
1	0	1	0	1	0		1	1	1	0	1	1	
1	1	0	1	1	0		0	1	1	1	1	1	
1	1	1	1	0	1		0	1	0	0	1	0	



Choose "next" LSb of Input Term 001 to Change

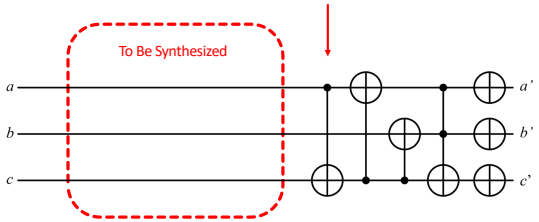
*D.M. Miller, D. Maslov and G.W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," in proc. IEEE/ACM Design Automation Conference (DAC), June 2-6, 2003.

28

Miller-Maslov-Dueck (MMD*) Method

Step 5: Change c^4 for Input Term 010 since a^3=1

a	b	c	a'	b'	c'		a^3	b^3	c^3	a^4	b^4	c^4
0	0	0	1	1	1		0	0	0	0	0	0
0	0	1	0	0	1		0	0	1	0	0	1
0	1	0	1	0	0		1	0	1	1	0	0
0	1	1	0	1	1		1	0	0	1	0	1
1	0	0	0	0	0		1	1	0	1	1	1
1	0	1	0	1	0		0	1	1	0	1	1
1	1	0	1	1	0		1	1	1	1	1	0
1	1	1	1	0	1		0	1	0	0	1	0



Choose LSb of Input Term 010 to Change

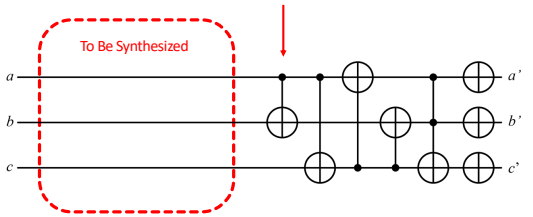
*D.M. Miller, D. Maslov and G.W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," in proc. IEEE/ACM Design Automation Conference (DAC), June 2-6, 2003.

29

Miller-Maslov-Dueck (MMD*) Method

Step 6: Change b^5 for Input Term 010 since a^4=1

a	b	c	a'	b'	c'		a^4	b^4	c^4	a^5	b^5	c^5
0	0	0	1	1	1		0	0	0	0	0	0
0	0	1	0	0	1		0	0	1	0	0	1
0	1	0	1	0	0		1	0	0	1	1	0
0	1	1	0	1	1		1	0	1	1	1	1
1	0	0	0	0	0		1	1	1	1	0	1
1	0	1	0	1	0		0	1	1	0	1	1
1	1	0	1	1	0		1	1	0	1	0	0
1	1	1	1	0	1		0	1	0	0	1	0



Choose "next" LSb of Input Term 010 to Change

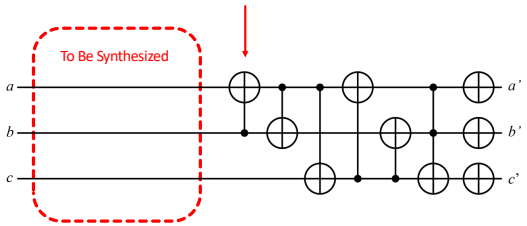
*D.M. Miller, D. Maslov and G.W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," in proc. IEEE/ACM Design Automation Conference (DAC), June 2-6, 2003.

30

Miller-Maslov-Dueck (MMD*) Method

Step 7: Change a^6 for Input Term 010 since b^5=1

a	b	c	a'	b'	c'		a^5	b^5	c^5	a^6	b^6	c^6
0	0	0	1	1	1		0	0	0	0	0	0
0	0	1	0	0	1		0	0	1	0	0	1
0	1	0	1	0	0		1	1	0	0	1	0
0	1	1	0	1	1		1	1	1	0	1	1
1	0	0	0	0	0		1	0	1	1	0	1
1	0	1	0	1	0		0	1	1	1	1	1
1	1	0	1	1	0		1	0	0	1	0	0
1	1	1	1	0	1		0	1	0	1	1	0



Choose "next" LSb of Input Term 010 to Change

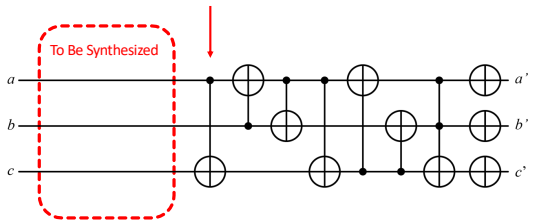
*D.M. Miller, D. Maslov and G.W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," in proc. IEEE/ACM Design Automation Conference (DAC), June 2-6, 2003.

31

Miller-Maslov-Dueck (MMD*) Method

Step 8: Change c^7 for Input Term 100 since a^6=1

a	b	c	a'	b'	c'		a^6	b^6	c^6	a^7	b^7	c^7
0	0	0	1	1	1		0	0	0	0	0	0
0	0	1	0	0	1		0	0	1	0	0	1
0	1	0	1	0	0		0	1	0	0	1	0
0	1	1	0	1	1		0	1	1	0	1	1
1	0	0	0	0	0		1	0	1	1	0	0
1	0	1	0	1	0		1	1	1	1	1	0
1	1	0	1	1	0		1	0	0	1	0	1
1	1	1	1	0	1		1	1	0	1	1	1



Choose LSb of Input Term 100 to Change

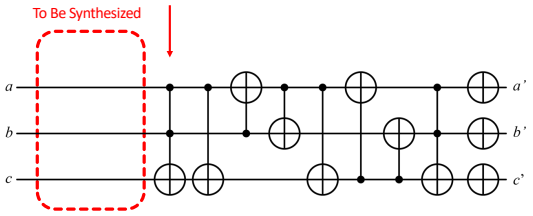
*D.M. Miller, D. Maslov and G.W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," in proc. IEEE/ACM Design Automation Conference (DAC), June 2-6, 2003.

32

Miller-Maslov-Dueck (MMD*) Method

Step 9: Change c^8 for Input Term 101 since a^7b^7=11

a	b	c	a'	b'	c'		a^7	b^7	c^7	a^8	b^8	c^8
0	0	0	1	1	1		0	0	0	0	0	0
0	0	1	0	0	1		0	0	1	0	0	1
0	1	0	1	0	0		0	1	0	0	1	0
0	1	1	0	1	1		0	1	1	0	1	1
1	0	0	0	0	0		1	0	0	1	0	0
1	0	1	0	1	0		1	1	0	1	1	1
1	1	0	1	1	0		1	0	1	1	0	1
1	1	1	1	0	1		1	1	1	1	1	0



Choose LSb of Input Term 101 to Change

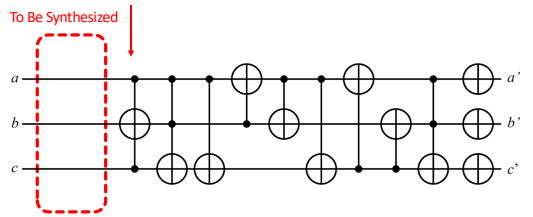
*D.M. Miller, D. Maslov and G.W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," in proc. IEEE/ACM Design Automation Conference (DAC), June 2-6, 2003.

33

Miller-Maslov-Dueck (MMD*) Method

Step 10: Change b^9 for Input Term 101 since a^8c^8=11

a	b	c	a'	b'	c'		a^8	b^8	c^8	a^9	b^9	c^9
0	0	0	1	1	1		0	0	0	0	0	0
0	0	1	0	0	1		0	0	1	0	0	1
0	1	0	1	0	0		0	1	0	0	1	0
0	1	1	0	1	1		0	1	1	0	1	1
1	0	0	0	0	0		1	0	0	1	0	0
1	0	1	0	1	0		1	1	1	1	0	1
1	1	0	1	1	0		1	0	1	1	1	1
1	1	1	1	0	1		1	1	0	1	1	0



Choose "next" LSb of Input Term 101 to Change

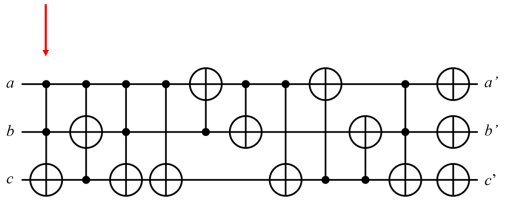
*D.M. Miller, D. Maslov and G.W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," in proc. IEEE/ACM Design Automation Conference (DAC), June 2-6, 2003.

34

Miller-Maslov-Dueck (MMD*) Method

Step 11: Change c^{10} for Input Term 110 since a^9b^9=11

a	b	c	a'	b'	c'		a^9	b^9	c^9	a^{10}	b^{10}	c^{10}
0	0	0	1	1	1		0	0	0	0	0	0
0	0	1	0	0	1		0	0	1	0	0	1
0	1	0	1	0	0		0	1	0	0	1	0
0	1	1	0	1	1		0	1	1	0	1	1
1	0	0	0	0	0		1	0	0	1	0	0
1	0	1	0	1	0		1	0	1	1	0	1
1	1	0	1	1	0		1	1	1	1	1	0
1	1	1	1	0	1		1	1	0	1	1	1



Synthesis Complete!!

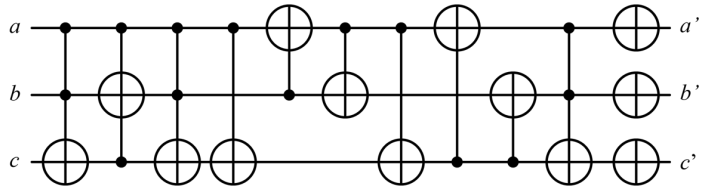
Choose LSb of Input Term 110 to Change

*D.M. Miller, D. Maslov and G.W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," in proc. IEEE/ACM Design Automation Conference (DAC), June 2-6, 2003.

35

Miller-Maslov-Dueck (MMD*) Method

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



*D.M. Miller, D. Maslov and G.W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," in proc. IEEE/ACM Design Automation Conference (DAC), June 2-6, 2003.

36

Miller-Maslov-Dueck (MMD*) Method

- This Basic Method Outputs Last Gate First and Iteratively Transforms the Output List into the Input List in Lexicographical Order
- Could Alternatively Output First Gate First and Iteratively Transform the Input List into the Output List
- "Bidirectional Method" Chooses Among the Two Approaches at Each Iteration
  - Can Yield a Smaller-cost Circuit
- MMD is Exponentially Expensive
  - Must Represent Entire Function – Exponential Size in Worst-case
  - Exponential Number of Iterations in Worst-case
- Not in the Form of Typical Oracle Structure: pre-image values are not present at oracle circuit outputs

*D.M. Miller, D. Maslov and G.W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," in proc. IEEE/ACM Design Automation Conference (DAC), June 2-6, 2003.

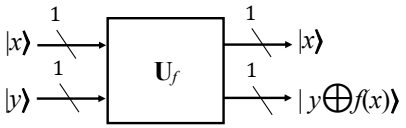
37

Why do QC Algorithm Designer's Care About Synthesis?

- For the Algorithms we have Studied, Many Such as Deutsch, Grover, Simon, QFT, Shor, have Embedded Functions
- Luckily, Most of these Embedded Functions have Unitary Representations
  - Analysis of the Functions Allow for a Unitary Representation
  - Exception is: Deutsch and Grover (and others)
- If We are Designing Quantum Algorithms, We Need Synthesis Methods to Embed Functions, Many of Which are Likely to be in Irreversible Form in there Native Format
- Let's Consider the Embedded Functions in the Algorithms that we have Studied so far in this Class

38

Deutsch's Algorithm Oracle

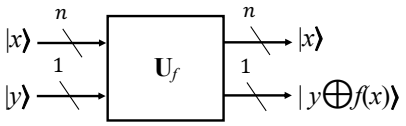


- $f(x)$ is a Single-bit Irreversible Switching Function "embedded" inside $\mathbf{U}_f$ "gate"
- Actual $f(x)$ embedded is unknown, but is Either Balanced or Imbalanced
- We Synthesized it in the Notes by Using Our Intuitive Knowledge of the Four Forms of Single-bit Switching Functions:

$$f(0)=0 \qquad f(0)=1 \qquad f(1)=0 \qquad f(1)=1$$

39

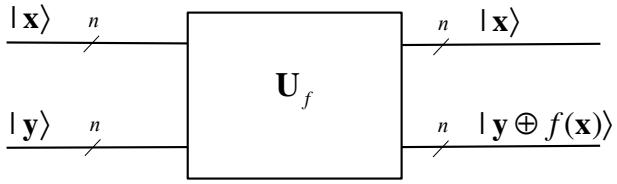
Grover's Algorithm Oracle



- $f(x)$ is Grover's Oracle Function that Indicates if the Search Object is Present in the $|\boldsymbol{x}\rangle$ Kets or Not
- The Embedded $f(x)$ Function is Single-bit Since it Indicates "yes" or "no" and it Must be Created when Implementing Grover's Search
- We Synthesized it in the Notes by Forming the Transfer Matrix as a Sum of Projectors, but this is Not Always Easy to Convert into Gates and it is exponentially complex

40

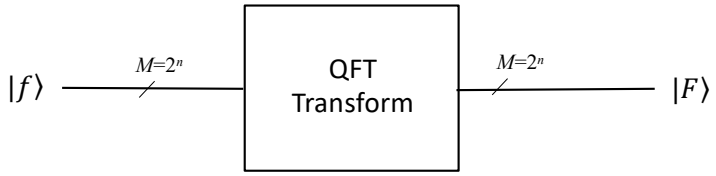
Simon's Periodicity Oracle



- $f(x)$ is an Embedded Function for Which We Desire to Determine the Periodicity
- We Completely Ignored the Problem of How to Embed the Function Into the Oracle in the Class Notes for Simon's Periodicity Algorithm

41

Quantum Fourier Transform Oracle



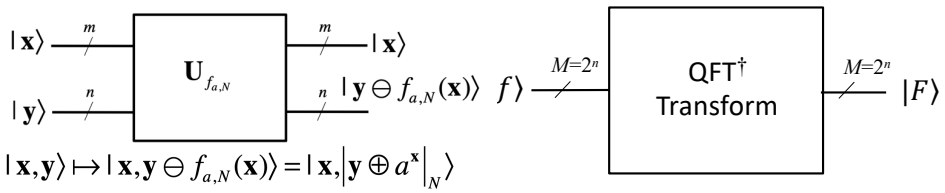
- QFT "oracle" is a Change of Basis of a Function f to Express it as F, Where:

$$|f\rangle = \sum_{j=0}^{M-1} f_j |j\rangle \qquad |F\rangle = \sum_{k=0}^{M-1} F_k |k\rangle \qquad F_k = \mathbf{DFT}(f_j) = \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} f_j e^{i\frac{2\pi jk}{M}}$$

- We Showed that the Binary Decomposition of the DFT Transform Matrix is Unitary and Mapped it Into Quantum Gates for the QFT

42

Shor's Factoring Oracle

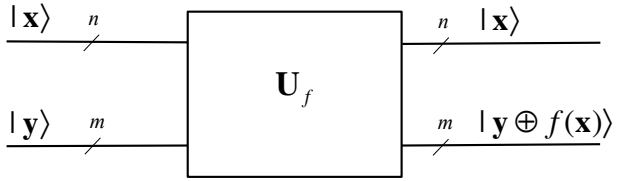


- QFT "oracle" Previously Described
- Modular Powers Oracle *Happened* to be Decomposed as a Cascade of Controlled Rotation Operators in a Radix-2 (bit-wise) Decomposition (we got Lucky!)

43

Typical Oracle Structure

- For the Algorithms we have Studied (Deutsch, Grover, Simon, QFT, Shor), there is an Embedded Function within an Oracle
- Most Oracles are of This Form:



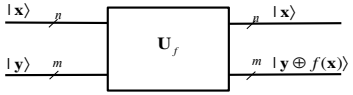
- QFT is the Exception, but it is Known that DFT is an Orthogonal Transform, so it is Really just a Projector that Changes the Basis of a Function
- The Question is: How do we, as QC Algorithm Designers, Embed Arbitrary Functions into Oracles of this Form?

44

Typical Oracle Structure (cont.)

- The Typical Structure is Useful Since the Ancilla Qubits, $|\boldsymbol{y}\rangle$, can be Used to Represent the Embedded Function (or its Inverse in Switching Function Form) Allowing the Function Pre-image Values, $|\boldsymbol{x}\rangle$, to be Entangled with Each Image Value, $|f\rangle$
  - This Entanglement has Proved Useful for "Filtering" Out Function Values (Image values) that we Wish to Discard
  - Especially When we Place Hadamards on the $|\boldsymbol{x}\rangle$ Pre-image Values to Allow the Embedded Function to be Evaluated at all Possible Domain Values Simultaneously
- We can Represent the Overall Transfer Matrix, $\mathbf{T}$, as a Sum of Projectors:

$$\mathbf{T} = |\boldsymbol{x}, \boldsymbol{y} \oplus f(\boldsymbol{x})\rangle\langle\boldsymbol{x}, \boldsymbol{y}|$$



- However, it is a Difficult Problem to Factor $\mathbf{T}$ into a Set of Product Matrices wherein the Product Matrices are Constrained to a Specific Set of Quantum Gates!

45

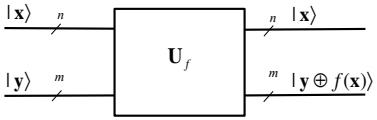
How can we Implement Oracles as QC Programmers?

- If Desired Function is in Closed Form (not likely), we can TRY to Manipulate it and Determine if we are Lucky Enough to Have a Unitary Form
  - But we Still Have to Implement it with Atomic Quantum Gate Operators
- In General, we will NOT have a Closed Form, we will have a Tabular Specification (*i.e*. Grover's Database Search, we Have a List of Data with Some Items to be Searched For Specified in a List)
- The Tabular Form is Likely to be an Irreversible Function
  - We can Use RTT to Convert it to a Reversible Form by Adding Ancilla/Garbage, but we still need to Synthesize it
- We can use MMD to Synthesize the Reversible Form (obtained from RTT), but it is Exponentially Complex AND it is NOT in the Form of a Typical Oracle

46

The ESOP Synthesis Method*

- This Technique was Invented in 2007 (here at SMU) and is the State-of-the-Art in Synthesizing Irreversible Functions into Oracles
- Input can be in the Form of a Tabular Listing that is Irreversible
  - Tabular Listings are Exponentially Complex (in space) Although Some Efficient Methods for Reducing Average Spatial Complexity can be used such as .pla files, PCN cube lists, and various Decision Diagrams
- ESOP Method Automatically Produces a Reversible Function without Requiring RTT
- ESOP Method Automatically Results in a "Typical Oracle" Structure



*K. Fazel, M.A. Thornton and J.E. Rice, "ESOP-based Toffoli Gate Cascade Generation," in proc. *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 206-209, August 22-24, 2007.

47

Exclusive-OR Sum-of-Products

- The ESOP Form* is a Subject in Switching Theory (Classical Digital Logic)
- Basic Undergraduate Intro. to Digital Logic Classes Typically Discuss Two Normal Forms of Switching Functions
  - The SOP – "Sum-of-Products" Form (aka, Minimized Sum-of-Minterms)
  - The POS – "Product-of-Sums" Form (aka, Minimized Product-of-Maxterms)
- We Briefly Review ESOP from the Point of View of Classical Switching Theory (Digital Circuits or Boolean Algebra)
- This ESOP Review will Provide an Intuitive Understanding of the ESOP Form
- We will then Show how the ESOP Form Relates to Reversible Logic

*In the Cryptography Community, ESOP Form is a Generalization of a Minimized Form known as the "Algebraic Normal Form" or ANF

48

Classical Sum-of-Minterms Form for Digital Logic

- Consider the Following Truth Table for a 3-Variable Single-output Function:

x_2	x_1	x_0	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

- The Switching Function can be written in symbolic form directly from the truth table as a "Sum-of-Minterms" (SOM):

$$f = \bar{x}_2\bar{x}_1x_0 + \bar{x}_2x_1\bar{x}_0 + x_2x_1\bar{x}_0$$

- We can use Boolean Algebra Theorems/Postulates to "Minimize" this into a "Sum-of-Products" (SOP):

$$\begin{aligned} f &= \bar{x}_2\bar{x}_1x_0 + \bar{x}_2(x_1\bar{x}_0) + x_2(x_1\bar{x}_0) \\ &= \bar{x}_2\bar{x}_1x_0 + (\bar{x}_2 + x_2)(x_1\bar{x}_0) \\ &= \bar{x}_2\bar{x}_1x_0 + x_1\bar{x}_0 \end{aligned}$$

- This is a "minimized" Sum-of-Products (SOP) form
  - "Minimized" in the sense that it has Fewer Product Terms and Literals

49

Minimizing SOP Forms

- You may have learned about "Karnaugh Maps" that are a way to Minimize SOP Forms without Using Algebra

x_2	x_1	x_0	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

- A K-map is an Alternative Arrangement of a Truth Table:

x_2 \ x_1x_0	00	01	11	10
0	0	1	0	1
1	0	0	0	1

- Circling Entries Allows for Directly Obtaining the Minimal SOP Form:

$$f = \bar{x}_2\bar{x}_1x_0 + x_1\bar{x}_0$$

50

Minimizing Switching Function SOP Forms

- You may have learned about "Karnaugh Maps" that are a way to Minimize SOP Forms without Using Algebra
- Minimizing SOPs is a Classic Problem in Complexity Theory, it is NP-Hard and Reduces to the "Set Covering" Problem
  - Determining if a Given Expression can be Further Minimized is NP-Complete
- In your Intro. to Digital Logic Class, you were asked to Minimize Functions, but they were so Small, you could easily find (one of the) minimal solutions
  - Using algebra OR using K-maps OR (perhaps) using other methods
- Modern Electronic Design (ICs) deal with Functions of 100's or even 1000's of Variables, no Human could minimize these
  - Minimizing is important because it means fewer transistors, faster circuits and less power is required
- Therefore, a LOT of Research has gone into finding Heuristics to Minimize Switching Functions

51

Minimizing Switching Function

- As Mentioned, there are Other Forms like POS (Product-of-Sums)
- Another Form that is Not Usually Taught in Intro. to Digital Logic is the "Exclusive-OR Sum-of-Products" or ESOP Form
- You may Remember that there are Two Types of "OR" Gates:
  - Inclusive-OR and Exclusive-OR
  - The Inclusive-OR is often just called an "OR Gate"
- SOP Forms use the Inclusive-OR to combine Products
  - In Logic Diagrams, {OR, AND, NOT} gates are all that are needed for SOP
  - In Logic Diagrams, {XOR, AND} gates are all that are needed for ESOP
- ESOP Forms use the Exclusive-OR to Combine Products
- ESOP and SOP are Related by the Following Relationships:

$$\bar{x}_i = 1 \oplus x_i \qquad x_i + x_j = x_i \oplus x_j \oplus x_i x_j$$

52

Converting SOP to ESOP

- Using these Relationships, we can Algebraically Convert SOP to ESOP

$$\bar{x}_i = 1 \oplus x_i \qquad\qquad x_i + x_j = x_i \oplus x_j \oplus x_i x_j$$

- One form of ESOP is the "Reed-Muller" Form where all Literals have the Same Polarity (so there are 2^n Different RM forms)
- The "Polarity-zero" (PPRM) form requires that no Literals be Complemented
- Consider the Previous SOM Example:

$$f = \bar{x}_2\bar{x}_1x_0 + \bar{x}_2x_1\bar{x}_0 + x_2x_1\bar{x}_0$$

- We can Algebraically Convert it to PPRM as:

$$\begin{aligned} f &= (\bar{x}_2\bar{x}_1x_0 \oplus \bar{x}_2x_1\bar{x}_0 \oplus \bar{x}_2\bar{x}_1x_0\bar{x}_2x_1\bar{x}_0) + x_2x_1\bar{x}_0 \\ &= (\bar{x}_2\bar{x}_1x_0 \oplus \bar{x}_2x_1\bar{x}_0 \oplus \bar{x}_2\bar{x}_1x_1x_0\bar{x}_0) + x_2x_1\bar{x}_0 \end{aligned}$$

- Note the third product in parentheses has the AND of two literals of opposite polarity (twice in fact) so it goes to zero – "disjoint" minterms

53

Converting SOP to ESOP (cont.)

- We can Algebraically Convert it to PPRM as:

$$\begin{aligned} f &= (\bar{x}_2\bar{x}_1x_0 \oplus \bar{x}_2x_1\bar{x}_0 \oplus \bar{x}_2\bar{x}_1x_0\bar{x}_2x_1\bar{x}_0) + x_2x_1\bar{x}_0 \\ &= (\bar{x}_2\bar{x}_1x_0 \oplus \bar{x}_2x_1\bar{x}_0 \oplus \bar{x}_2\bar{x}_1x_1x_0\bar{x}_0) + x_2x_1\bar{x}_0 \end{aligned}$$

- Note the third product in parentheses has the AND of two literals of opposite polarity (twice in fact) so it goes to zero – "disjoint" minterms
- In fact, all minterms are disjoint from one another
- A minterm is a special case of a "product" term that contains all possible literals
- In general, two product terms may NOT be disjoint, but minterms always are
- If two Product Terms ARE Disjoint, it means they each contain at least one Literal of the same variable, but in opposite polarity
- If product Term p_i and p_j are Disjoint, then $p_i + p_j = p_i \oplus p_j$, if the two product terms are NOT disjoint, then $p_i + p_j = p_i \oplus p_j \oplus p_i p_j$

54

Converting SOP to ESOP (cont.)

- Given the Disjointness Observation, let's Start over with our Example using the following Identities to convert the example SOM into a PPRM form of ESOP: $\bar{x}_i = 1 \oplus x_i$ $x_i + x_j = x_i \oplus x_j \oplus x_i x_j$

$$f = \bar{x}_2\bar{x}_1x_0 + \bar{x}_2x_1\bar{x}_0 + x_2x_1\bar{x}_0$$
$$= \bar{x}_2\bar{x}_1x_0 \oplus \bar{x}_2x_1\bar{x}_0 \oplus x_2x_1\bar{x}_0$$

disjoint minterms

$$= (1 \oplus x_2)(1 \oplus x_1)x_0 \oplus (1 \oplus x_2)x_1(1 \oplus x_0) \oplus x_2x_1(1 \oplus x_0)$$
$$= (1 \oplus x_2 \oplus x_1 \oplus x_2x_1)x_0 \oplus x_1(1 \oplus x_2 \oplus x_0 \oplus x_2x_0) \oplus x_2x_1 \oplus x_2x_1x_0$$
$$= x_2x_1x_0 \oplus x_2x_0 \oplus x_1x_0 \oplus x_0 \oplus x_2x_1x_0 \oplus x_2x_1 \oplus x_1x_0 \oplus x_1 \oplus x_2x_1 \oplus x_2x_1x_0$$

- This looks like a mess, but we can use two properties of the XOR to simplify, these are $p_i \oplus p_i = 0$ and $p_i \oplus p_i \oplus p_i = p_i$.
- In fact, an "even" number of XORed terms goes to zero and an "odd" number of XORed terms reduces to a single instance of that term:

$$\bigoplus_{i=1}^{2n} (p) = 0 \qquad \bigoplus_{i=1}^{2n+1} (p) = p$$

55

Converting SOP to ESOP (cont.)

- Continuing with our conversion of SOM to PPRM-form of ESOP, given the XOR Relationships of "even" versus "odd" numbers of like terms:

$$f = \bar{x}_2\bar{x}_1x_0 + \bar{x}_2x_1\bar{x}_0 + x_2x_1\bar{x}_0$$
$$= x_2x_1x_0 \oplus x_2x_0 \oplus x_1x_0 \oplus x_0 \oplus x_2x_1x_0 \oplus x_2x_1 \oplus x_1x_0 \oplus x_1 \oplus x_2x_1 \oplus x_2x_1x_0$$
$$= x_2x_1x_0 \oplus x_2x_0 \oplus x_1 \oplus x_0$$

- Thus, we have obtained the ESOP form known as PPRM
- The Reed-Muller forms allow any particular variable to be either complemented, or not complemented, but not both. In Switching Theory language, we say that all literals must be "unate."
- Other ESOP forms allow Literals to be present in both complemented and uncomplemented form. In Switching Theory language, we would say these variables are allowed to be present in a "binate" form.

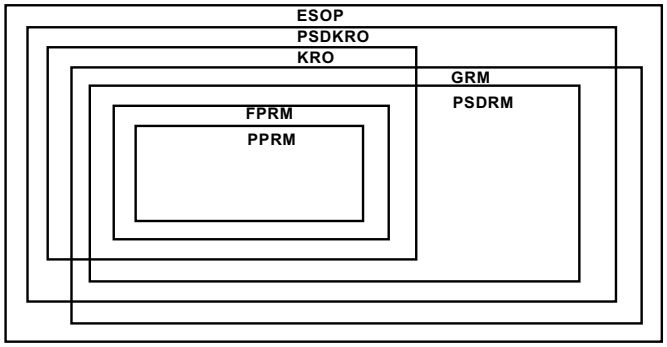
56

Different Classes of ESOP and other Theory

- Over the Years, many different subsets of ESOP forms have been defined with various ways to automatically derive them.
- There is rich theory behind these classes, but we will not provide that here.
- The backup section contains a summary of the different forms.
- It is also the case that the 2^n Reed-Muller Forms are in fact Discrete Fourier Transforms over the SOM form with associated Orthogonal Transformation Matrices. This is also detailed in the backup slides.
  - Because the RM forms are Fourier Transforms of SOM, corresponding "fast" transform algorithms are possible with butterfly diagrams – this is in the backup slides.
- Further, it is the Case that Many different forms of Decision Diagrams are Simply Graphical versions of different Classes of ESOPs. Again, we will not go into this theory.

57

Different Classes of ESOP



ESOP – Exclusive-OR Sum of Products

PSDKRO – Psuedo-Kronecker

KRO – Kronecker Form

GRM – Generalize Reed-Muller

Pseudo-Reed-Muller

FPRM – Fixed Polarity Reed-Muller

PPRM – Positive Polarity Reed-Muller

58

Examples of ESOP Forms

• PPRM (All Literals are Positive Polarity, or Polarity=0)

$$f = xy \oplus yz \oplus zx$$

• FPRM (All Literals have Same Polarity, or Polarity=2)

$$f = x\overline{y} \oplus \overline{y}z \oplus zx$$

• PSDRM (Some Literals have both Positive and Negative Polarity)

$$f = xy \oplus \overline{y}z \oplus \overline{z}x$$

• PSDKRO (Some Literals have both Positive and Negative Polarity and Cannot be Achieved Using KRO Type Expansions Only)

$$f = \overline{x} \oplus xy \oplus x\overline{y}$$

59

Examples of ESOP Forms (CONT.)

• GRM (Some Literals have both Positive and Negative Polarity and Cannot be Achieved Using PSDKRO Type Expansions Only)

$$f = x \oplus y \oplus \overline{x}\,\overline{y}$$

• KRO (Not GRM since two Product Terms with Same Set of Variables)

$$f = x\,y\,z \oplus \overline{x}\,\overline{y}\,\overline{z}$$

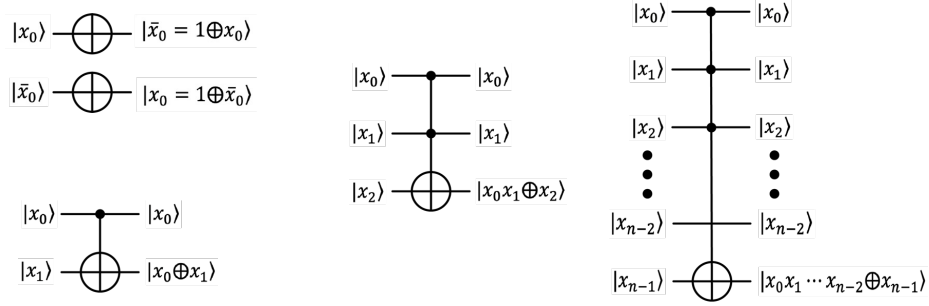
• General ESOP (Not GRM or PSDKRO)

$$f = x \oplus y \oplus x\,y \oplus \overline{x}\,\overline{y}$$

60

How is ESOP Relevant to Quantum Oracle Synthesis?

- It Turns out that Reversible Logic Gates can be Considered in Terms of the ESOP Switching Relationships as Shown:
  - Note that this result Assumes Qubits are in Their Computational Basis States Only (i.e., it is a "reversible logic" technique, not a general quantum method)



61

The ESOP Synthesis Method

- If an Irreversible Switching Function is Specified as a List of Pre-image and Image Values, then the List can be Transformed into an ESOP Representation
- Transforming them into an ESOP List Causes Each Term to Either be Disjoint in Relation to the Others, OR, it Causes Overlapping Bits to be Present an ODD Number of Times
- This Means that Each Pre-image/Image Pair Can be Inserted into a Toffoli Cascade (in any order), Thus Realizing the Irreversible Function as a Toffoli Cascade
- There are Certain Structural Considerations as Explained in the Following Slides
- Many Times, Switching Functions in this Form are Minimized in SOP Form, it is Absolutely Essential that the Irreducible Function be Transformed into ESOP Form BEFORE Mapping to a Circuit

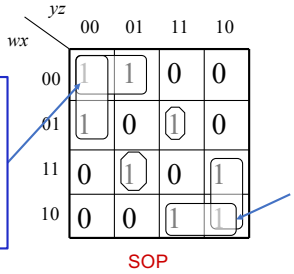
62

Why ESOP Rather Than SOP Form is Required

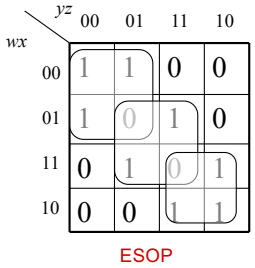
- Clearly, we wish to Minimize the Function Listing BEFORE Mapping to Reduce Quantum Cost; However, the Minimizer MUST Minimize to ESOP Rather than SOP Form

• Consider the function $f(x,y,w,z) = \Sigma(0,1,4,7,10,11,13,14)$

This Term is Covered Twice, an EVEN Number of Times!!! Mapping to Reversible Logic would cause it to go to Zero!!! Incorrect Circuit!!!!!



SOP



ESOP

In Digital Logic, the SOP form requires 6 AND gates
- Two 4-input and Four 3-input gates

In Digital Logic, the ESOP form only requires 3 AND gates
- Three 2-input gates

63

ESOP Mapping Algorithms

- As Mentioned, Converting from an SOP to a Minimized ESOP is an NP-hard Problem!!
- Thus, we MUST Rely on Heuristic Methods to Perform this Mapping
- Fortunately, the Switching Theory Community has Worked on this Problem since the 1980's
- Switching Theorists were Interested in ESOP Minimizers for Classical Logic Because it is Postulated that (on average) ESOP Minimized Switching Functions Require Fewer Literals than SOP Minimized Functions
  - Thus, in Electronic Circuits, there will be fewer transistors, faster circuits (sometimes), and less power dissipation
- One of the Best Heuristic Minimizers is EXORCISM4* by Alan Mishchenko (primary author) and Bob Brayton (UC Berkeley), although Others Exist

*A. Mishchenko and M.A. Perkowski, "Fast Heuristic Minimization of Exclusive-Sums-of-Products," in proc. 5th *International Reed-Muller Workshop* (RMW), pp. 242-250, https://pdxscholar.library.pdx.edu/ece_fac/195/, (last accessed April 24, 2022), 2001.

64

Switching Theory Interest in ESOP

- Hypothesis: On Average ESOP Requires Fewer Terms than SOP
- Exact Results for 5-Variable Functions (2^{32} total):
  - SOP – 7.46 Product Terms
  - ESOP – 6.16 Product Terms

EXAMPLE

$$f = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5$$

SOP: *16* Products, ESOP: *5* Products

$$f = x_1x_2 + x_3x_4 + \ldots + x_{2n-1}x_{2n}$$

SOP: *n* Products, ESOP: 2^n-*1* Products

65

ESOP Minimizers

- Prior to the 2007 Publication of the ESOP Reversible Logic Mapping Method*, ESOP Minimization was a Relatively Obscure Topic of Interest Only to a few Members in the Switching Theory Community
- Since then, Widespread Interest in ESOP Minimizers, and Many New Results have been Published
- Some Improvements in the 2007 Method, but they are Typically Minor and Special-case; the 2007 Method Remains State-of-the-Art in 2022
- Some New Approaches to ESOP Minimization have Occurred, but EXORCISM4 Remains one of the Best Approaches
- Following Slide Contains some Earlier ESOP Minimizers

*K. Fazel, M.A. Thornton and J.E. Rice, "ESOP-based Toffoli Gate Cascade Generation," in proc. *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 206-209, August 22-24, 2007.

66

Other ESOP Minimizers

- Extensions to QM Tabulation Method
  - Odd/Even Cube Covering [Thornton, et. al, '01]
- Rule Based Approaches
  - EXMIN2 [Sasao'93]
  - MINT [Kozlowski, et al. '95]
  - CANNES/EXORCISM [Perkowski, et al. '92]
  - EXORCISM4 [Mishchenko/Perkowski, et al. '01]
- Graph Based Approaches
  - XORDDs [Roy, et al'97]
  - Parity-BDDs [Meinel, et. al '98]
  - PSDKRO [Drechsler, et. al '98]

*K. Fazel, M.A. Thornton and J.E. Rice, "ESOP-based Toffoli Gate Cascade Generation," in proc. *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 206-209, August 22-24, 2007.

67

ESOP-based Toffoli Gate Cascade Generation*

- Based on EXORCISM4
- Very Large Circuits Generated
- Very Fast Synthesis Computer Runtime
- Based on:
  - Recursive Divide and Conquer Algorithm
  - Heuristic Cost Function
- *n*+*m* Qubits Required for *n*-input and *m*-output function

*K. Fazel, M.A. Thornton and J.E. Rice, "ESOP-based Toffoli Gate Cascade Generation," in proc. *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 206-209, August 22-24, 2007.

68

Approach*

- Utilize Toffoli Gate Analogy to AND/XOR
- Take Advantage of Classical ESOP Minimizers
- Generate a Cascade of 1 Gate per Resultant Cube
- Order of Quantum Gates DOES NOT Matter
  - Can Rearrange Order and Use Quantum NOT to Generate Inverse Control Qubit Polarity
  - Can Use Heuristic Cost Function to Determine Gate Order

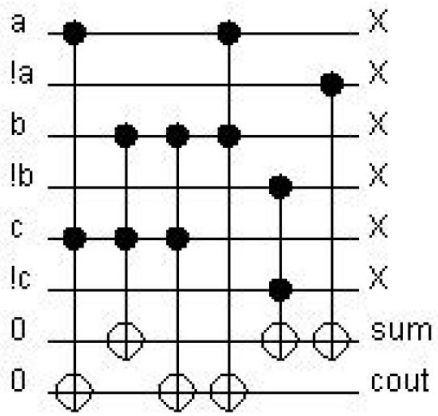
*K. Fazel, M.A. Thornton and J.E. Rice, "ESOP-based Toffoli Gate Cascade Generation," in proc. *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 206-209, August 22-24, 2007.

69

Full Adder Mapping with 2*n*+*m**

.pla File Format after ESOP Minimized

```
.i 3
.o 2
.type esop
1-1 01
-11 11
11- 01
-00 10
0-- 10
.e
```



*K. Fazel, M.A. Thornton and J.E. Rice, "ESOP-based Toffoli Gate Cascade Generation," in proc. *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 206-209, August 22-24, 2007.

70

Optimizations*

- Variables that Appear in Single Polarity Allow Alternate Qubit to be Removed
- Variables Appearing in Both Polarities:
  - Realize all Gates in One Polarity First and Second Polarity after Insertion of Quantum Not Gate
  - Reduces from 2 Qubits to 1 Qubit
- Must Determine Order of Variables in Which to Insert Quantum NOT Gates to Minimize Number:
  - Use Heuristic Based on Merit/Cost Metric

*K. Fazel, M.A. Thornton and J.E. Rice, "ESOP-based Toffoli Gate Cascade Generation," in proc. *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 206-209, August 22-24, 2007.

71

QC NOT Insertion*

- Criteria for NOT Gate Insertion
- Insert NOT First For Qubits that:
  1. Are Balanced in Polarity (occur in approximately Equal Numbers of Polarity in a Group of Cubes)
  2. Occur Most Frequently in a Group of Cubes
- Qubit First in Order has 1 QC NOT Gate, 2nd in Order 2 QC NOT Gates, *n*th in Order 2^n NOT Gates
- Try to Do Polarity Splitting Such that Remaining Sublists do not Contain Both Polarities

*to appear, Fazel, Thornton, Rice PACRIM' 07

72

Merit/Cost Function*

- Definitions:

$$v_i = \begin{cases} +1, & \text{variable in cube } i \text{ is positive} \\ 0, & \text{variable in cube } i \text{ is don't care} \\ -1, & \text{variable in cube } i \text{ is negative} \end{cases}$$

- Scaling Constants $\alpha,\beta \in [0,1]$, $\beta = 1-\alpha$
- Merit/Cost Function:

$$\text{cost}_v = \alpha\left(\frac{1}{\sum |v_i|}\right) + \beta\left(\left|\sum v_i\right|\right)$$

*K. Fazel, M.A. Thornton and J.E. Rice, "ESOP-based Toffoli Gate Cascade Generation," in proc. *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 206-209, August 22-24, 2007.

73

Algorithm*

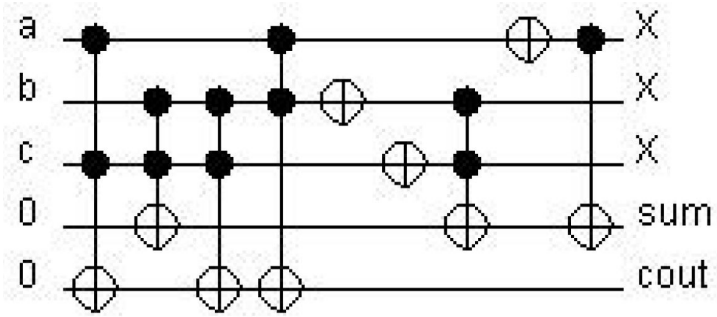
```
reorder(cubes, polarity, vars)
    if(cubes.isEmpty || vars.isEmpty)
      return cubes
    bestVar = calcBestVar(cubes, vars)
    {pCubes, nCubes} = split(cubes, bestVar)
    pReorder = reorder(pCubes, positive, vars – bestVar)
    nReorder = reorder(nCubes, negative, vars – bestVar)
    if(polarity==negative)
      nReorderNots = addNots(nReordered, bestVar)
    reorderedCubes = reconnect(pReorder, nReorderNots)
    return reorderedCubes

cascadeGen(esop)
    reorderedCubes = reorder(esop.cubes, positive, esop.vars)
    cascade = convertCubesToToffoli(reorderedCubes)
    removeExtraNots(cascade)
```

*K. Fazel, M.A. Thornton and J.E. Rice, "ESOP-based Toffoli Gate Cascade Generation," in proc. *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 206-209, August 22-24, 2007.

74

Full Adder after Algorithm*



*K. Fazel, M.A. Thornton and J.E. Rice, "ESOP-based Toffoli Gate Cascade Generation," in proc. *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 206-209, August 22-24, 2007.

75

(2007-era) Experimental Results*

Circuit	in	out	esopCubes	TOF Gates	NOT Gates	α	β	esopTime (s)	cascadeGenTime (s)
bw	5	28	22	251	11	1.00	0.00	0.02	0.00
xor5	5	1	5	5	2	1.00	0.00	0.00	0.00
5xp1	7	10	31	61	29	0.50	0.50	0.01	0.00
risc	8	31	27	133	19	0.50	0.50	0.02	0.00
cordic	23	2	776	1546	711	0.75	0.25	14.67	0.02
ttt2	24	21	60	92	41	1.00	0.00	0.08	0.00
vg2	25	8	184	214	286	1.00	0.00	0.11	0.02
bc0	26	11	167	562	158	0.75	0.25	0.47	0.00
in7	26	10	35	64	34	0.75	0.25	0.02	0.00
chkn	29	7	144	147	202	0.50	0.50	0.14	0.02
term1	34	10	540	702	127	0.25	0.75	2.62	0.02
apex2	39	3	1637	1755	1005	0.75	0.25	45.27	0.06
seq	41	35	248	1877	272	0.25	0.75	1.29	0.02
apex1	45	45	288	1348	306	0.75	0.25	1.31	0.03
apex3	54	50	258	2045	278	0.75	0.25	6.90	0.05
dalu	75	16	1472	3472	644	1.00	0.00	0.11	0.00
e64	65	65	65	129	64	0.75	0.25	0.06	0.03
example2	85	66	205	280	81	0.25	0.75	3.35	0.05
x4	97	71	299	460	155	1.00	0.00	2.41	0.05
apex5	117	88	398	541	163	0.50	0.50	4.98	0.16
ex4	128	28	316	321	417	0.75	0.25	0.91	0.20
apex6	135	99	409	569	236	1.00	0.00	6.86	0.14
frg2	143	139	1116	1971	339	0.50	0.50	184.70	0.72
i2	201	1	257	257	536	0.75	0.25	0.80	0.28

*K. Fazel, M.A. Thornton and J.E. Rice, "ESOP-based Toffoli Gate Cascade Generation," in proc. *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 206-209, August 22-24, 2007.

76

Toffoli Mapping Conclusions*

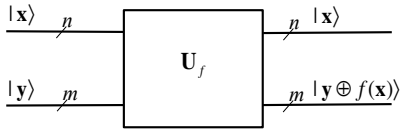
- Technique is Very Fast and Can Handle Relatively Large Circuits
- Does Not Require Exponentially Sized Truth Table as Initial Input
- Viable Initial Mapping Procedure for Further Synthesis Optimization Methods

*K. Fazel, M.A. Thornton and J.E. Rice, "ESOP-based Toffoli Gate Cascade Generation," in proc. *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 206-209, August 22-24, 2007.

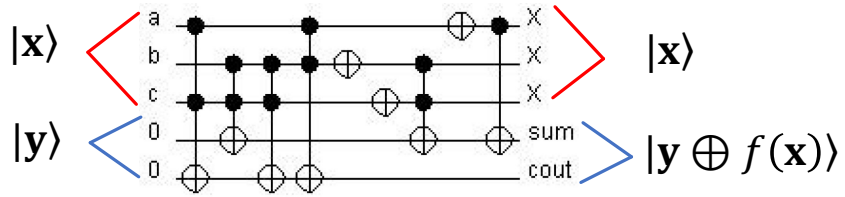
77

ESOP Method Produces Oracle Structure

- Method Natively Produces Oracle Structure & Inherently Converts from Irreversible to Reversible Form



Single-bit Full-adder Embedded in a Quantum Oracle Produced by ESOP Method*



*K. Fazel, M.A. Thornton and J.E. Rice, "ESOP-based Toffoli Gate Cascade Generation," in proc. *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 206-209, August 22-24, 2007.

78

Recent Research Results

- J.M. Henderson, E.R. Henderson, A. Sinha, M.A. Thornton and D.M. Miller, Automated Quantum Oracle Synthesis with a Minimal Number of Qubits, SPIE 12517, Quantum Information Science, Sensing, and Computation XV, April 30-May 4, 2023, 1251706 (18 pp.), June 13, 2023.
- A. Sinha, E.R. Henderson, J.M. Henderson, E.C. Larson and M.A. Thornton, A Programmable True Random Number Generator using Commercial Quantum Computers, SPIE 12517, Quantum Information Science, Sensing, and Computation XV, April 30-May 4, 2023, 1251705 (15 pp.), June 13, 2023.
- A. Sinha, E.R. Henderson, J.M. Henderson and M.A. Thornton, Automated Quantum Memory Compilation with Improved Dynamic Range, International Conference for High Performance Computing, Networking, Storage, and Analysis (SC22), International Workshop on Quantum Computing software (QSC22), November 13, 2022, 14 pp., arXiv version.

79

BACKUP SLIDES

80

Expansion Types

Shannon Expansion:

$$f(x_n, x_{n-1}, .., x_i, ., x_1) = \overline{x}_i \overline{f}_0 \oplus x_i f_0$$

Positive Davio Expansion:

$$f(x_n, x_{n-1}, .., x_i, ., x_1) = f_0 \oplus x_i f_2$$

Negative Davio Expansion:

$$f(x_n, x_{n-1}, .., x_i, ., x_1) = f_1 \oplus \overline{x}_i f_2$$

Function Co-factors:

$$f_0 = f(x_n, x_{n-1}, .., x_i = 0, ., x_1)$$
$$f_1 = f(x_n, x_{n-1}, .., x_i = 1, ., x_1)$$
$$f_2 = f_0 \oplus f_1$$

81

Classes of ESOP Forms

• Forms are Classified by their ***EXPANSION TYPE*** (Shannon, positive-Davio, negative Davio and the ***POLARITY*** of the Literals

• *PPRM* – Two-Level form with all Literals Uncomplemented and Uses the Logical AND and XOR Operators. 1 FPRM for Every Function – Canonical RM Form.

• *FPRM* – Two-Level Form that has Each Literal (for a given variable) Either Complemented or Uncomplemented, but not Both. Uses the Logical AND and XOR Operations. Binary Number Formed by Assigning a "1" to x_i if it is Complemented and a "0" to x_j if it is Uncomplemented Gives the Polarity Number. 2^n FPRMs for Each Function.

82

Classes of ESOP Forms (CONT.)

- *KRO* – Two-Level Form Where the Function is Expanded by Either the Positive Davio OR the Negative Davio OR the Shannon Expansion Everywhere. This is a Generalization of FPRM. 3^n KRO Forms for Each Function.
- *PSDRM* – Two-Level Form Where the Function is Expanded by Either Positive Davio OR Negative Davio only. 2^{2^n-1} Different PSDRM Forms for Each Function.
- *PSDKRO* – Two-Level Form Where Each Expansion may be One of Positive Davio , Negative Davio or Shannon. Various Co-factors are Expanded in Any one of these Three ways. Different Functions Possible. 3^{2^n-1}

83

Examples of ESOP Forms

- PPRM (All Literals are Positive Polarity)

$$f = xy \oplus yz \oplus zx$$

- FPRM (All Literals have Same Polarity – Polarity=2)

$$f = x\bar{y} \oplus \bar{y}z \oplus zx$$

- PSDRM (Some Literals have both Positive and Negative Polarity)

$$f = xy \oplus \bar{y}z \oplus \bar{z}x$$

- PSDKRO (Some Literals have both Positive and Negative Polarity and Cannot be Achieved Using KRO Type Expansions Only)

$$f = \bar{x} \oplus xy \oplus x\bar{y}$$

84

Examples of ESOP Forms (CONT.)

• GRM (Some Literals have both Positive and Negative Polarity and Cannot be Achieved Using PSDKRO Type Expansions Only)

$$f = x \oplus y \oplus \overline{x}\,\overline{y}$$

• KRO (Not GRM since two Product Terms with Same Set of Variables)

$$f = x\,y\,z \oplus \overline{x}\,\overline{y}\,\overline{z}$$

• ESOP (Not GRM or PSDKRO)

$$f = x \oplus y \oplus x\,y \oplus \overline{x}\,\overline{y}$$

85

Positive Polarity Reed-Muller (PPRM)

$$f = a_0 \oplus a_1 x_1 \oplus \ldots \oplus a_n x_n \oplus a_{12} x_1 x_2 \oplus a_{13} x_1 x_3 \oplus \ldots.$$
$$\oplus\, a_{nn-1} x_n x_{n-1} \oplus \ldots \oplus a_{12\ldots n} x_1 x_2 \ldots x_n$$

$a_i \in \{0,1\}$

x_i *are all positve polarity* (*uncomplemented literals*)

EXAMPLE

x_3	x_2	x_1	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

$$f = \sum(m_1, m_2, m_6) = \overline{x}_3\,\overline{x}_2\,x_1 + \overline{x}_3\,x_2\,\overline{x}_1 + x_3\,x_2\,\overline{x}_1$$

• Sum of Minterms is Disjoint:

$$f = \overline{x}_3\,\overline{x}_2\,x_1 \oplus \overline{x}_3\,x_2\,\overline{x}_1 \oplus x_3\,x_2\,\overline{x}_1$$

• Can Use the Following Identities:

$$1 \oplus x = x$$
$$x \oplus x = 0$$
$$x \oplus x \oplus x = x$$

86

PPRM Example (cont)

$$f = \bar{x}_3\,\bar{x}_2\,x_1 \oplus \bar{x}_3\,x_2\,\bar{x}_1 \oplus x_3\,x_2\,\bar{x}_1$$

$$f = (1 \oplus x_3)(1 \oplus x_2)x_1 \oplus (1 \oplus x_3)x_2(1 \oplus x_1) \oplus x_2x_3(1 \oplus x_1)$$

$$f = (1 \oplus x_2 \oplus x_3 \oplus x_2x_3)x_1 \oplus (1 \oplus x_1 \oplus x_3 \oplus x_1x_3)x_2 \oplus x_2x_3 \oplus x_1x_2x_3$$

$$f = x_1 \oplus x_1x_3 \oplus x_1x_2 \oplus x_1x_2x_3 \oplus x_2 \oplus x_2x_3 \oplus x_1x_2 \oplus x_1x_2x_3 \oplus x_2x_3 \oplus x_1x_2x_3$$

x_3	x_2	x_1	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

$$f = x_1 \oplus x_2 \oplus x_1x_3 \oplus x_1x_2x_3$$

General Form:

$$f = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_3 \oplus a_{12}x_1x_2 \oplus a_{13}x_1x_3 \oplus a_{23}x_2x_3 \oplus a_{123}x_1x_2x_3$$

$$(a_0, a_1, a_2, a_{12}, a_3, a_{13}, a_{23}, a_{123}) = (0,1,1,0,0,1,0,1)$$

87

Fixed Polarity Reed-Muller (PPRM)

$$f = a_0 \oplus a_1\dot{x}_1 \oplus a_2\dot{x}_2 \oplus a_3\dot{x}_3 \oplus a_{12}\dot{x}_1\dot{x}_2 \oplus a_{13}\dot{x}_1\dot{x}_3 \oplus a_{23}\dot{x}_2\dot{x}_3 \oplus a_{123}\dot{x}_1\dot{x}_2\dot{x}_3$$

$a_i \in \{0,1\}$

$\dot{x}_i$ *are all either Complemented or Uncomplemented*

EXAMPLE

$$(\dot{x}_1, \dot{x}_2, \dot{x}_3) = (\bar{x}_1, x_2, \bar{x}_3) \rightarrow Polarity - 5$$

$$f = \bar{x}_3\,\bar{x}_2\,x_1 + \bar{x}_3\,x_2\,\bar{x}_1 + x_3\,x_2\,\bar{x}_1 = \bar{x}_3\,\bar{x}_2\,x_1 \oplus \bar{x}_3\,x_2\,\bar{x}_1 \oplus x_3\,x_2\,\bar{x}_1$$

$$f = \bar{x}_3\,(1 \oplus x_2)(1 \oplus \bar{x}_1) \oplus \bar{x}_3\,x_2\,\bar{x}_1 \oplus (1 \oplus \bar{x}_3)\,x_2\,\bar{x}_1$$

$$f = \bar{x}_3\,(1 \oplus x_2 \oplus \bar{x}_1 \oplus x_2\bar{x}_1) \oplus \bar{x}_3\,x_2\,\bar{x}_1 \oplus x_2\bar{x}_1 \oplus \bar{x}_3\,x_2\,\bar{x}_1$$

$$f = \bar{x}_3 \oplus \bar{x}_3x_2 \oplus \bar{x}_1\bar{x}_3 \oplus x_2\bar{x}_1 \oplus \bar{x}_3\,x_2\,\bar{x}_1$$

$$f = a_0 \oplus a_1\bar{x}_1 \oplus a_2x_2 \oplus a_3\bar{x}_3 \oplus a_{12}\bar{x}_1x_2 \oplus a_{13}\bar{x}_1\bar{x}_3 \oplus a_{23}x_2\bar{x}_3 \oplus a_{123}\bar{x}_1x_2\bar{x}_3$$

$$(a_0, a_1, a_2, a_{12}, a_3, a_{13}, a_{23}, a_{123}) = (0,0,0,1,1,1,1,1)$$

88

Canonical Forms (PPRM)

Disjunctive Sum of Minterms

$$f = m_0\bar{x}_1\bar{x}_2\bar{x}_3 + m_1\bar{x}_1\bar{x}_2x_3 + m_2\bar{x}_1x_2\bar{x}_3 + m_3\bar{x}_1x_2x_3$$
$$+ m_4x_1\bar{x}_2\bar{x}_3 + m_5x_1\bar{x}_2x_3 + m_6x_1x_2\bar{x}_3 + m_7x_1x_2x_3$$

Exclusively Disjunctive Sum of Minterms

$$f = m_0\bar{x}_1\bar{x}_2\bar{x}_3 \oplus m_1\bar{x}_1\bar{x}_2x_3 \oplus m_2\bar{x}_1x_2\bar{x}_3 \oplus m_3\bar{x}_1x_2x_3$$
$$\oplus m_4x_1\bar{x}_2\bar{x}_3 \oplus m_5x_1\bar{x}_2x_3 \oplus m_6x_1x_2\bar{x}_3 \oplus m_7x_1x_2x_3$$

Positive Polarity Reed-Muller Form

$$f = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_3 \oplus a_{12}x_1x_2 \oplus a_{13}x_1x_3 \oplus a_{23}x_2x_3 \oplus a_{123}x_1x_2x_3$$

Fixed Polarity Reed-Muller Form

$$f = a_0 \oplus a_1\dot{x}_1 \oplus a_2\dot{x}_2 \oplus a_3\dot{x}_3 \oplus a_{12}\dot{x}_1\dot{x}_2 \oplus a_{13}\dot{x}_1\dot{x}_3 \oplus a_{23}\dot{x}_2\dot{x}_3 \oplus a_{123}\dot{x}_1\dot{x}_2\dot{x}_3$$

m or a vectors Uniquely Specify a Function

$$m_i \in \{0,1\} \qquad a_i \in \{0,1\}$$

89

Relationship Between Canonical Forms

Operation	SOP	ESOP
Multiplicative	AND	AND
Additive	OR	XOR

- Relationships Between the Two:

$$a \bullet b = a \bullet b$$
$$a + b = a \oplus b \oplus ab$$
$$\bar{a} = 1 \oplus a$$

- Consider All Binary Functions of $n=1$ Variable

$2^{2^1} = 4$ *Functions*: $f(x) = d_0\bar{x} + d_1x$ *where* $di \in \{0,1\}$

$f_0 = 0 \qquad f_1 = \bar{x}$

$f_2 = x \qquad f_3 = 1$

90

Relationship Between Canonical Forms (cont)

Operation	SOP	ESOP
Multiplicative	AND	AND
Additive	OR	XOR

$$a \bullet b = a \bullet b$$
$$a + b = a \oplus b \oplus ab$$
$$\overline{a} = 1 \oplus a$$

$$f(x) = d_0\overline{x} + d_1 x \quad where \quad d_i \in \{0,1\}$$

$$f(x) = d_0(1 \oplus x) \oplus d_1 x$$

$$f(x) = d_0 \oplus x(d_0 \oplus d_1)$$

Let:

$$c_0 \equiv d_0$$
$$c_1 \equiv d_0 \oplus d_1$$

Then:

$$f(x) = c_0 \oplus c_1 x$$

91

The Reed-Muller Transform

$$f(x) = d_0\overline{x} + d_1 x \quad where \quad d_i \in \{0,1\}$$
$$f(x) = c_0 \oplus c_1 x$$

- Relationship Between the d_i and c_i constants

$$\begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \end{bmatrix}$$

- This is the Polarity-0 Reed-Muller Transform for $n=1$ Variable
- For More than 1 Variable, the *Kronecker* Product Can Be Used:

$$G_n = \begin{bmatrix} G_{n-1} & 0 \\ G_{n-1} & G_{n-1} \end{bmatrix}$$

- Recursive Definition Allows for Usage of "fast"-transform techniques
  - "Butterfly" Diagrams like FFT
- Must Use the Multiplication and Addition Operations as Defined when Deriving the Transform (• and $\oplus$)

92

PPRM Transform

$$f = \sum(m_1, m_2, m_6) = \bar{x}_3\,\bar{x}_2\,x_1 + \bar{x}_3\,x_2\,\bar{x}_1 + x_3\,x_2\,\bar{x}_1$$

x_3	x_2	x_1	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

- First Compute the Transformation Matrix for $n=3$ Variables Using Kronecker Expansion

$$G_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \qquad G_3 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

$$G_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \qquad G_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

93

Calculating the PPRM Spectrum

x_3	x_2	x_1	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

$$f = \sum(m_1, m_2, m_6) = \bar{x}_3\,\bar{x}_2\,x_1 + \bar{x}_3\,x_2\,\bar{x}_1 + x_3\,x_2\,\bar{x}_1$$

$$f = x_1 \oplus x_2 \oplus x_1x_3 \oplus x_1x_2x_3$$

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_{12} \\ a_3 \\ a_{13} \\ a_{23} \\ a_{123} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

- Same as Algebraic Result!

94

Polarity-5 FPRM Transform

$$f = \sum(m_1, m_2, m_6) = \overline{x}_3\,\overline{x}_2\,x_1 + \overline{x}_3\,x_2\,\overline{x}_1 + x_3\,x_2\,\overline{x}_1$$

x_3	x_2	x_1	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

- First Compute the Transformation Matrix for $n=3$ Variables Using Kronecker Expansion

$$G_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \qquad G_3 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}^{\overline{x}_3} \otimes \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{x_2} \otimes \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}^{\overline{x}_1}$$

$$G_3 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \qquad G_3 = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

95

Calculating the Polarity-5 FPRM Spectrum

x_3	x_2	x_1	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

$$f = \sum(m_1, m_2, m_6) = \overline{x}_3\,\overline{x}_2\,x_1 + \overline{x}_3\,x_2\,\overline{x}_1 + x_3\,x_2\,\overline{x}_1$$

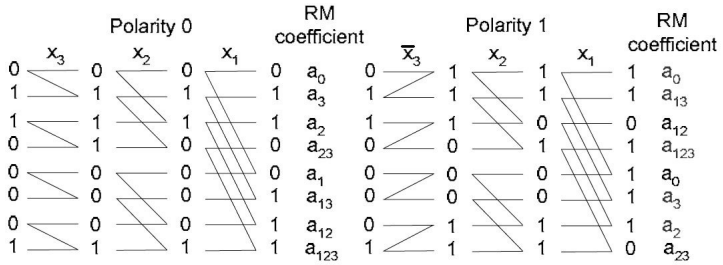
$$f = \overline{x}_1 \oplus x_2\,\overline{x}_3 \oplus \overline{x}_1 x_2 \oplus \overline{x}_1\overline{x}_3 \oplus \overline{x}_1 x_2 \overline{x}_3$$

$$\begin{bmatrix} a_{13} \\ a_3 \\ a_{123} \\ a_{23} \\ a_1 \\ a_0 \\ a_{12} \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

- Same as Algebraic Result!

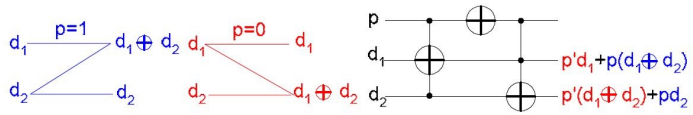
96

RM Transform Butterfly Diagrams



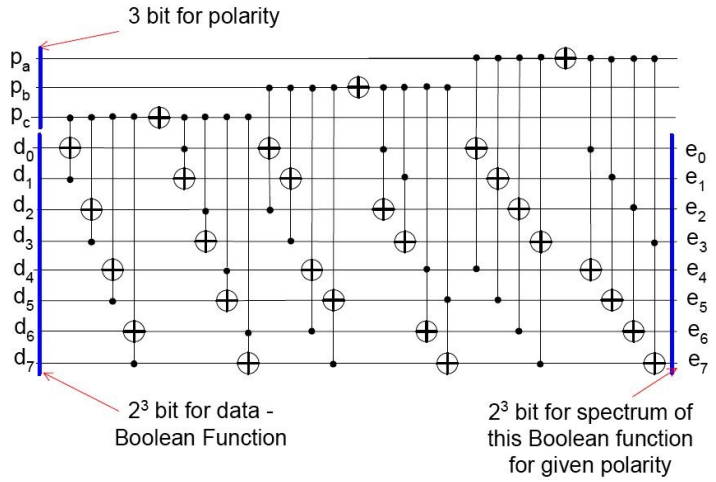
97

Quantum Circuit for RM Transform



98

Generalized QC for RM Transform



99

Classical Sum-of-Minterms Form for Digital Logic

- Consider the Following Truth Table for a 3-Variable Single-output Function:

x_2	x_1	x_0	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

- The Switching Function can be written in symbolic form directly from the truth table as a "Sum-of-Minterms" (SOM):

$$f = \bar{x}_2\bar{x}_1x_0 + \bar{x}_2x_1\bar{x}_0 + x_2x_1\bar{x}_0$$

- We can use Boolean Algebra Theorems/Postulates to "Minimize" this into a "Sum-of-Products" (SOP):

$$f = m_0\bar{x}_2\bar{x}_1\bar{x}_0 + m_1\bar{x}_2\bar{x}_1x_0 + m_2\bar{x}_2x_1\bar{x}_0 + m_3\bar{x}_2x_1x_0 + m_4x_2\bar{x}_1\bar{x}_0 + m_5x_2\bar{x}_1x_0 + m_6x_2x_1\bar{x}_0 + m_6x_2x_1x_0$$

- sd

100