BINARY REVERSIBLE BACKTRACKING

Programming for Synthesis of Reversible Logic Circuits using Backtracking and Search

Prashanth kumar Miryala
Lavanya Jayaram
Narendra Metta
Sricharan kaza
Padma Vetcha

Outline

- Introduction to Reversible Logic
- Previous work in this area
- Our approach
- Algorithm
- Future work

Reversible Logic

- Why do we need reversible logic
- What is reversible logic
- What are its applications

Laundauer's Principle

It states that logic computation that are not reversible necessarily generate *kT ln 2* Joules of heat energy for every bit of information energy that is lost

Reversible Logic

- A digital combinational logic circuit is reversible if it maps each input pattern to a unique output pattern
- A reversible logic design will not result in information loss so this avoids the unwanted heat generated.
- It is proved that using traditional irreversible logic gates necessarily leads to power dissipation regardless of underlying technology. For power not to be dissipated in an arbitrary circuit, it must be built from reversible gates.

Reversible gates

- Not
- CNOT (Feynman) gate
- Toffoli gate
- Fredkin gate

Gate Type	Functionality	Gate Notation
1*1 Not	$x^+ = \overline{x}$	NOT(x)
2*2 Feynman [4]	$x^+ = x$ $y^+ = x \oplus y$	FEY(x,y)
3*3 Toffoli [19]	$x^+ = x$ $y^+ = y$ $z^+ = xy \oplus z$	TOF3(x,y,z)
3*3 Fredkin [5]	$x^+ = x$ $y^+ = \overline{x}y \oplus xz$ $z^+ = \overline{x}z \oplus xy$	FRE(x,y,z)

Applications of reversible logic

- Low power CMOS design
- Optical computing
- Quantum computing
- Nanotechnologies

Previous works done on the reversible logic synthesis

- Khlopotine, Perkowski, Kerntopf
- Maslov, Miller and Dueck

Work of khlopotine and Perkowski

- In their paper they introduce the the generalised family of Feynman, Toffoli and Fredkin gates (shown in fig 1, 2, 3)
- Here function f1 denotes any arbitrary boolean function of one variable and f2 denotes any arbitrary boolean function of two variables.
- It is obsereved, that all these gates are reversible and can be extended to gates with an arbitrary no. of control inputs.

Generalised Families of reversible gates

Fig. 1. Generalized Feynman Gate

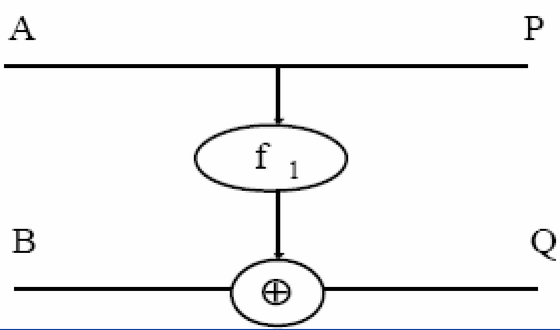


Fig. 2. Generalized Toffoli Gate

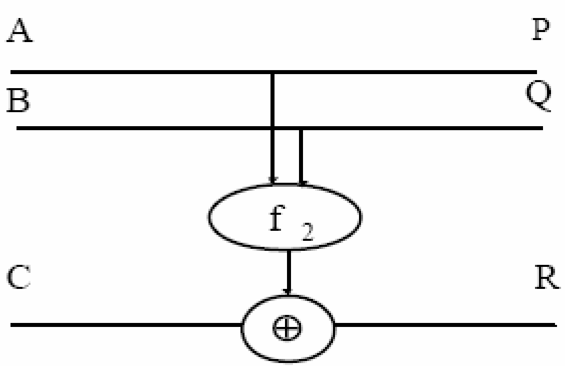
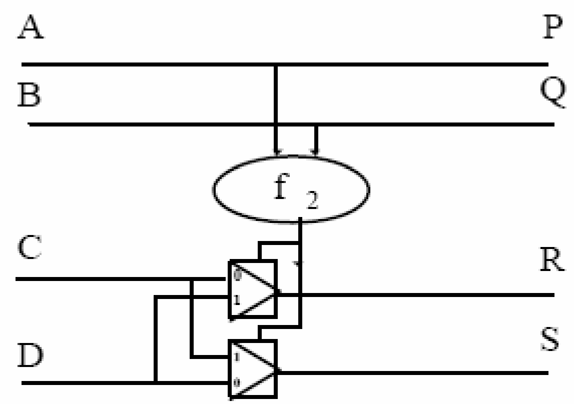


Fig. 3. Generalized Fredkin Gate



Compositional Synthesis Methods

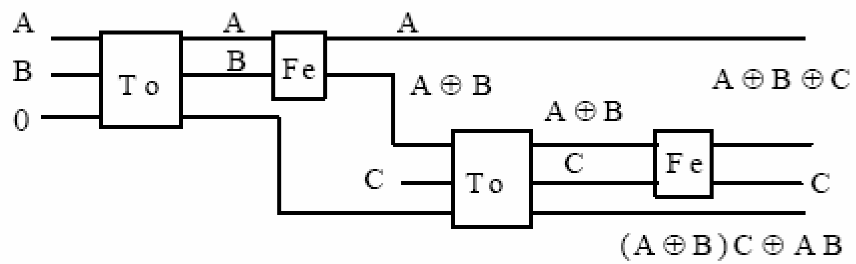
- It is seen that the simplest structure for composition are cascades.
- Single output functions can be realized using reversible gates.
- Realization of Adder circuit is shown
- It has two primary outputs, two potential garbage outputs, three primary inputs and 1 constant input.

Synthesis method (cont'd.)

- A heuristic used here, is maximization of combination of coincidence count and minimization of entropy.
- By doing so they reduced the cost function.
- Considering a more general case, they realized a full adder using Toffoli and Feynman gates.

Adder Circuit realised using compostion

Fig. 6. Full Adder realized using Composition



Contribution

- The main contribution of their approach is an introduction of new reversible gate families and convergent synthesis algorithms for single-output functions.
- Also their method uses information-theory-based cost function, called coincidence-count.

Disadvantages

- It is applicable to small and medium-sized benchmarks.

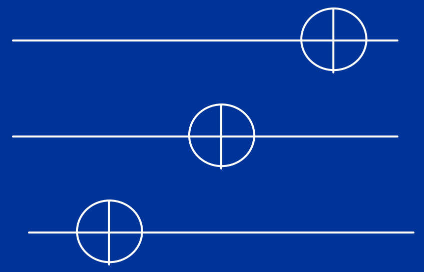
Works of Maslov *et al*

- He uses a transformation based algorithm using Toffoli, Swap, Fredkin and inverter gates
- This resulted in circuits with fewer gates than the previous works

Synthesis using Maslov approach

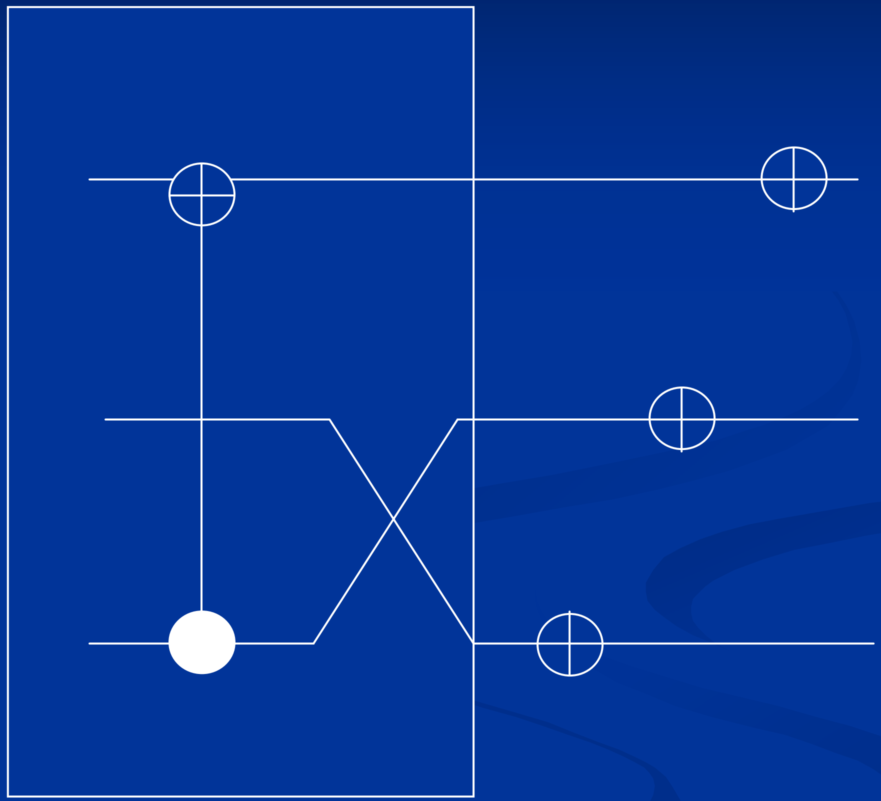
Input	Output
000	111
001	001
010	100
011	011
100	000
101	010
110	110
111	101

Circuit



Step-1

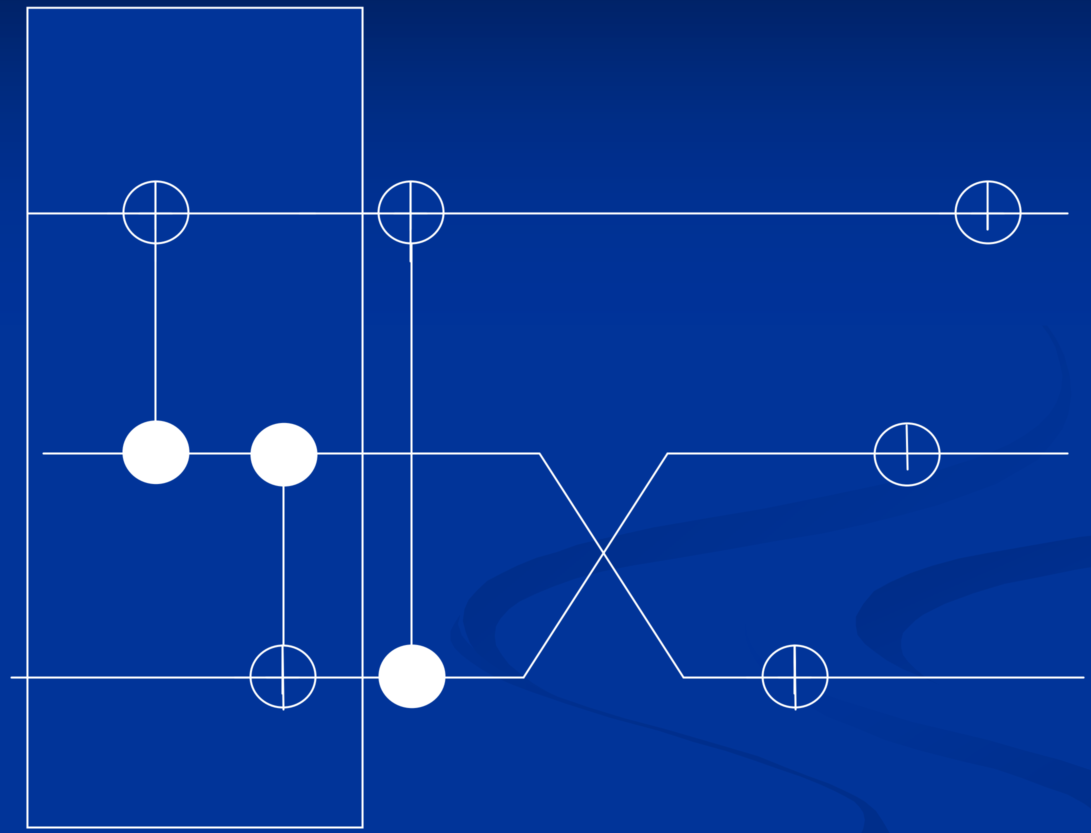
Input	Output	step-1
000	111	000
001	001	110
010	100	011
011	011	100
100	000	111
101	010	101
110	110	001
111	101	010



Step 2

Step-2

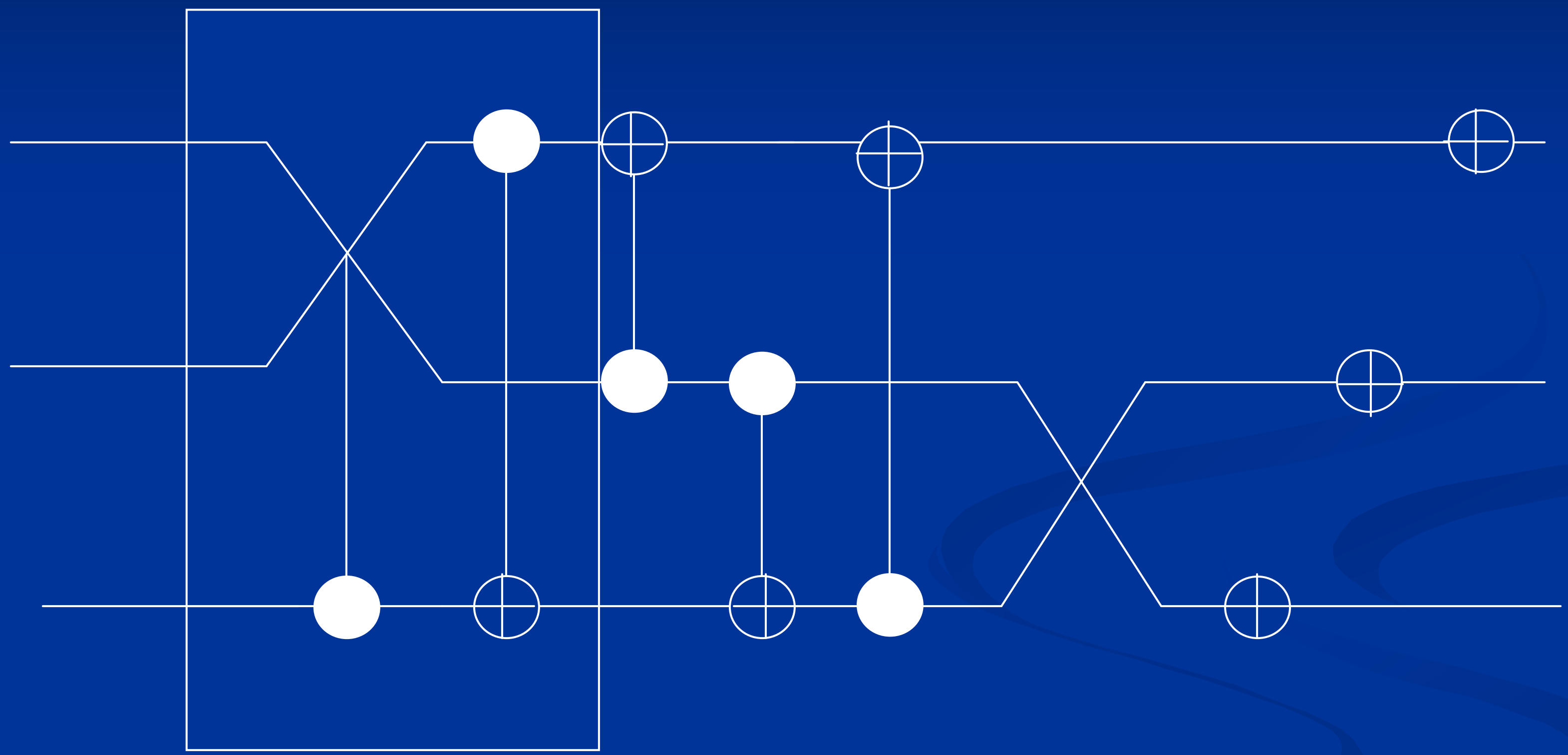
Input	Output	step-1	step-2
000	111	000	000
001	001	110	001
010	100	011	111
011	011	100	100
100	000	111	011
101	010	101	110
110	110	001	010
111	101	010	101



Step 3

Step-3

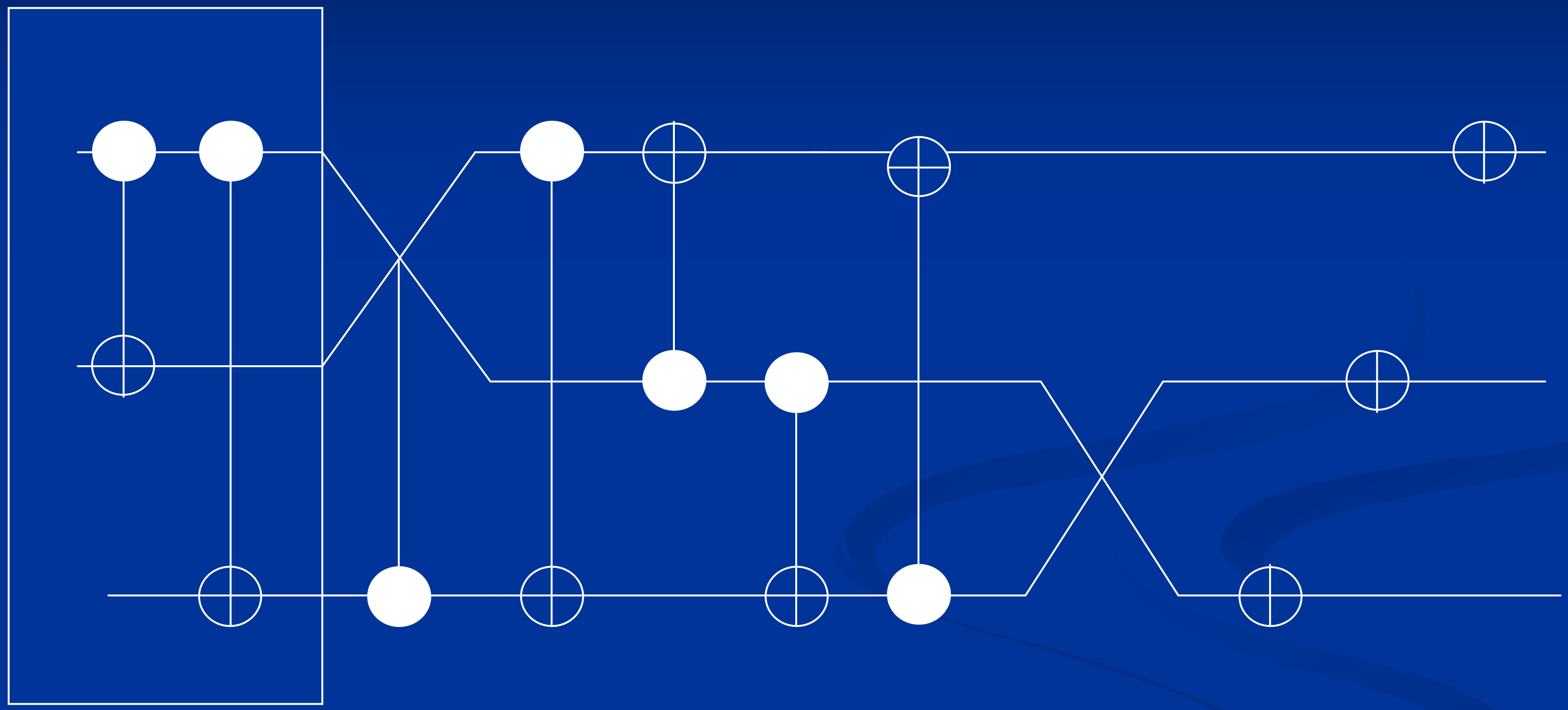
Input	Output	step-1	step-2	step-3
000	111	000	000	000
001	001	110	001	001
010	100	011	111	010
011	011	100	100	100
100	000	111	011	110
101	010	101	110	011
110	110	001	010	111
111	101	010	101	101



Step 4

Step-4

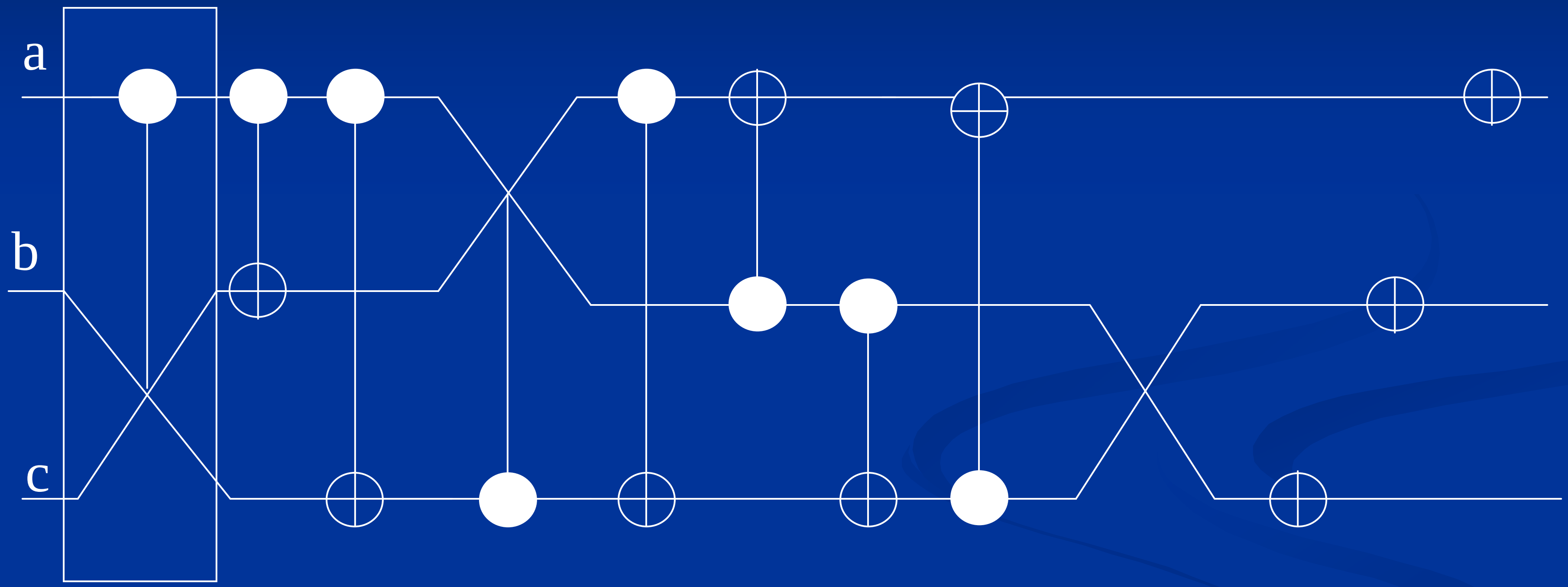
Input	Output	step-1	step-2	step-3	step-4
000	111	000	000	000	000
001	001	110	001	001	001
010	100	011	111	010	010
011	011	100	100	100	011
100	000	111	011	110	111
101	010	101	110	011	101
110	110	001	010	111	110
111	101	010	101	101	100



Step 5

Step-5

Input	Output	step-1	step-2	step-3	step-4	step-5
000	111	000	000	000	000	000
001	001	110	001	001	001	001
010	100	011	111	010	010	010
011	011	100	100	100	011	011
100	000	111	011	110	111	100
101	010	101	110	011	101	110
110	110	001	010	111	110	101
111	101	010	101	101	100	111

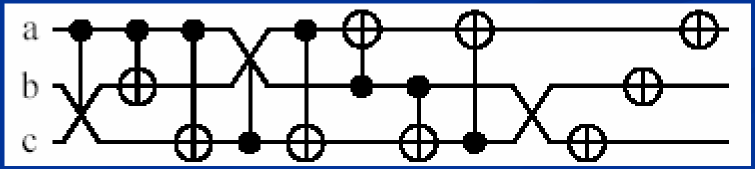


Step 6

Step-6

IN	Out	S1	S2	S3	S4	S5	S6
000	111	000	000	000	000	000	000
001	001	110	001	001	001	001	001
010	100	011	111	010	010	010	010
011	011	100	100	100	011	011	011
100	000	111	011	110	111	100	100
101	010	101	110	011	101	110	101
110	110	001	010	111	110	101	110
111	101	010	101	101	100	111	111

In	Out	S0	S1	S2	S3	S4	S5
000	111	**000**	**000**	**000**	**000**	**000**	**000**
001	001	110	**001**	**001**	**001**	**001**	**001**
010	100	011	111	**010**	**010**	**010**	**010**
011	011	100	100	100	**011**	**011**	**011**
100	000	111	011	110	111	**100**	**100**
101	010	101	110	011	101	110	**101**
110	110	001	010	111	110	101	**110**
111	101	010	101	101	100	111	**111**
apply gates:		$T(a)$ $T(b)$ $T(c)$	$F(b,c)$ $T(c;a)$	$T(b;c)$ $T(b;a)$	$T(a;c)$ $F(c;a,b)$	$T(a;c)$ $T(a;b)$	$F(a;b,c)$



Advantages of Maslov’s approach

- It is simple greedy algorithm
- It works in all cases
- It terminates in all cases

Disadvantages

- We don’t end up with an optimal solution
- If we deal with variables more than 3 then we need toffoli gates with that many inputs, this is bad

Bidirectional modification

- The basic algorithm can be applied in both directions of the cascade. Then a network is constructed fom the two ends simultaneously by growing the number of gates from two sides.
- Such an algorithm on average will converge faster.

Our plan of Work

- We plan to synthesize reversible functions with n=4 , where n is the number of inputs
- Synthesis will be done backwards, that is building the circuit from the given output specification to obtain the Identity function.
- Further, Bi-directional approach can also be equipped to obtain better results

Possible Approaches

- Exhaustive search.
- Selecting a gate from a library of gates which reduce the hamming distance in each step .
- New heuristic methods.
- Minimization of PPRM literals.

Our approach

- We have a gate library from which we choose a particular gate
- List of gates in the library

    1.Not

    2.Feynman

    3.Toffoli

    4.Fredkin

    5.Swap

- We choose a gate in such a way that it reduces the hamming distance between the input and the output truth tables .

Description

- We start constructing the circuit from the Output specification.
- Assuming the narrowest gate in the cascade, we evaluate the output specification.
- The best gate position (less hamming dt.) is chosen.
- Thus each gate is placed and the cost is evaluated for the best combination.

EXAMPLE FUNCTION

Input	Output
a	a
b	$a \oplus ad \oplus b$
c	c
d	$acd \oplus bc \oplus d$

abcd
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

Input Side

abcd
0000
0001
0010
0011
0100
0101
0111
0110
1100
1001
1110
1010
1000
1101
1011
1111

Output Side

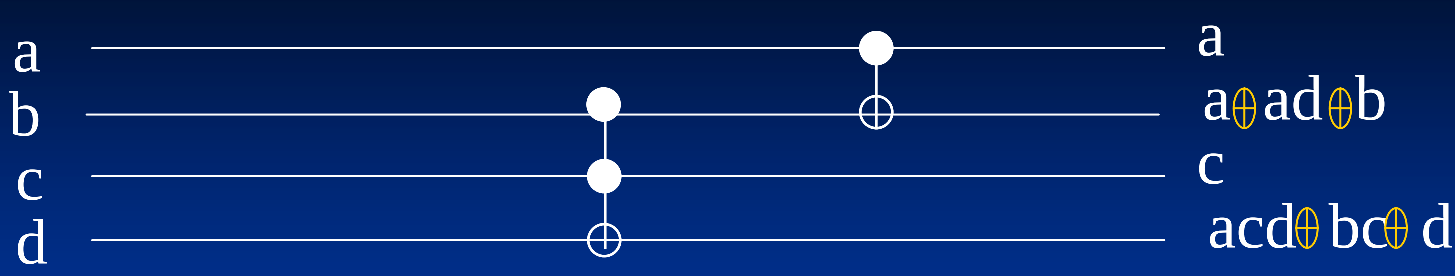
Input Side (abcd)	Step 1 (abcd)	Output Side (abcd)
0000	0000	0000
0001	0001	0001
0010	0010	0010
0011	0011	0011
0100	0100	0100
0101	0101	0101
0110	0111	0111
0111	0110	0110
1000	1000	1100
1001	1101	1001
1010	1010	1110
1011	1110	1010
1100	1100	1000
1101	1001	1101
1110	1111	1011
1111	1011	1111

Input Side

Step 1

Hamming dt=8

Output Side



Input Side (abcd)	Step 2 (abcd)	Step 1 (abcd)	Output Side (abcd)
0000	0000	0000	0000
0001	0001	0001	0001
0010	0010	0010	0010
0011	0011	0011	0011
0100	0100	0100	0100
0101	0101	0101	0101
0110	0110	0111	0111
0111	0111	0110	0110
1000	1000	1000	1100
1001	1101	1101	1001
1010	1010	1010	1110
1011	1111	1110	1010
1100	1100	1100	1000
1101	1001	1001	1101
1110	1110	1111	1011
1111	1011	1011	1111

Input Side

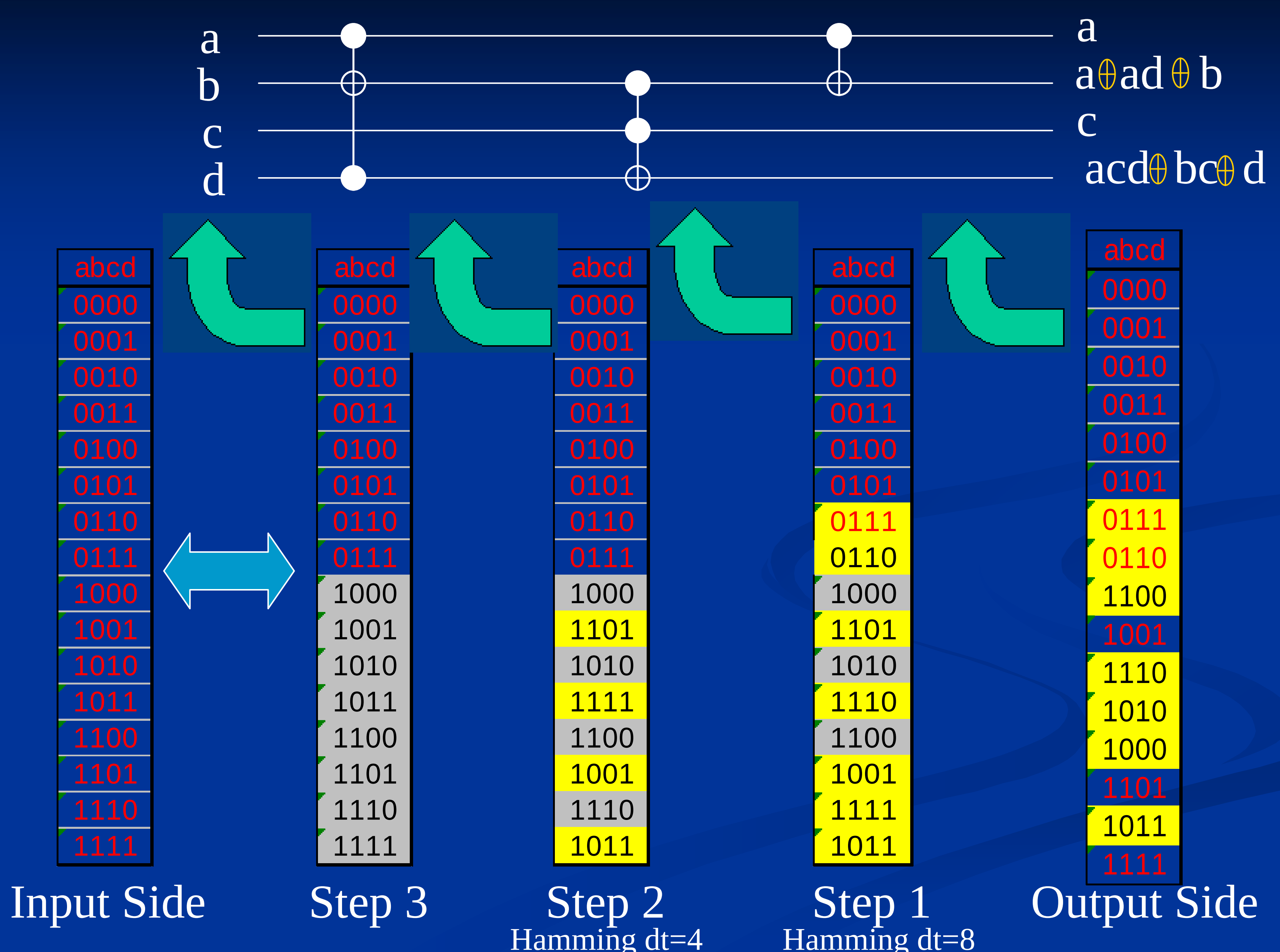
Step 2

Hamming dt=4

Step 1

Hamming dt=8

Output Side


a
b
c
d
a
a⊕ad⊕b
c
acd⊕bc⊕d
abcd
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
abcd
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
abcd
0000 0001 0010 0011 0100 0101 0110 0111 1000 1101 1010 1111 1100 1001 1110 1011
abcd
0000 0001 0010 0011 0100 0101 0111 0110 1000 1101 1010 1110 1100 1001 1111 1011
abcd
0000 0001 0010 0011 0100 0101 0111 0110 1100 1001 1110 1010 1000 1101 1011 1111
Input Side
Step 3
Step 2
Hamming dt=4
Step 1
Hamming dt=8
Output Side

Algorithm

- Step 1: Select a gate from the library which reduces the hamming distance to the maximum extent between input and output.
- Step 2: If two or more gates reduce the hamming distance equally

  then go to step 4

  else go to step 3

Algorithm cont'd

- Step 3: If the hamming distance is zero

  then Exit.

  else go to step 1.

- Step 4:We find a next best gate for each of the previous gate and we choose the best pair of gates. If there is a tiebreak we choose the first pair of gates

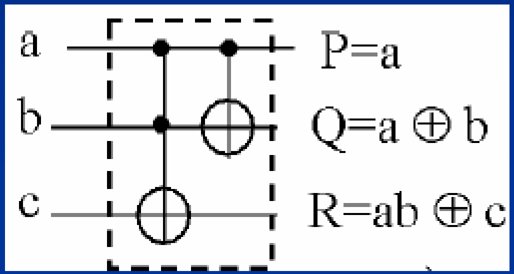
  then go to step 3.

Backtracking

- A method used in search algorithms to retreat from an unacceptable position and restart the search at a previously known "good" position.
- When two gates reduce the hamming distance equally in a particular stage, we use a Look ahead strategy and try to predict the best gate for our cascade.

Possible ways of improving the result

- Introduce new kind of gates in gate library.
- A new heuristic considering both hamming distance and PPRM literals minimization, while selecting a gate.



Peres gate

THANK YOU!!!