Moving Forward: A Non-Search Based Synthesis Method towards Efficient CNOT-Based Quantum Circuit Synthesis Algorithms

Mehdi Saeedi, Morteza Saheb Zamani, Mehdi Sedighi

Email: {msaeedi, szamani, msedighi}@ aut.ac.ir

Quantum Design Automation Lab, Computer Engineering Department

Amirkabir University of Technology

Tehran, Iran

ASPDAC 2008

Outline

- Introduction
- Basic Concept
- Previous Work
- Synthesis Algorithm (MOSAIC)
- Experimental Results
- Future Works
- Conclusions

Quantum Computing

- The fundamental limits of CMOS technology
- The enormous amount of required processing power for future applications
- New computational models
- Quantum computing

Synthesis

- Quantum information processing is in the preliminary state
- No mature synthesis method for quantum circuit synthesis has been proposed yet
- A systematic algorithm for Boolean reversible circuit synthesis

Boolean Reversible Functions

- n-input, n-output,
- Maps each input assignment to a unique output assignment
- Example: a 3-input, 3-output function (0,1,2,7,4,5,6,3)

AND

a_0	a_1	a_2	f_0	f_1	f_2	F
0	0	0	0	0	0	0
0	0	1	0	0	1	1
0	1	0	0	1	0	2
0	1	1	1	1	1	7
1	0	0	1	0	0	4
1	0	1	1	0	1	5
1	1	0	1	1	0	6
1	1	1	0	1	1	3

Power dissipation

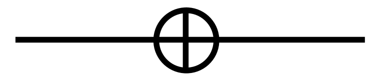
- R. Landauer in IBM Journal, 1961
  - Every lost bit causes an energy loss
  - When a computer erases a bit of information, the amount of energy dissipated into the environment is at least $k_B T \ln 2$
- C. Bennett, IBM Journal, 1973
  - To avoid power dissipation in a circuit, the circuit must be built with reversible gates

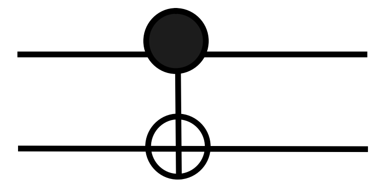
Applications of reversible circuits

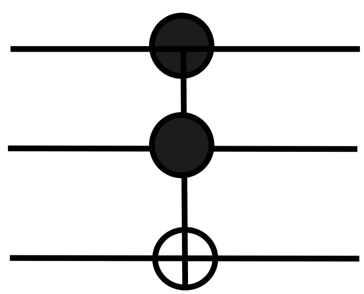
- Low power CMOS design
  - Reversible 4-bit adder
    - “A reversible carry-look-ahead adder using control gates”, *Integration, the VLSI Journal*, vol. 33, pp. 89-104, 2002
    - 384 transistors with **no power rails**
- Optical computing
- Quantum computing
  - Each unitary quantum gate is intrinsically reversible

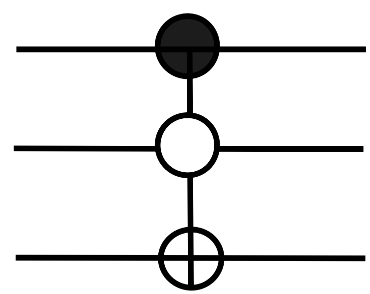
Basic Concept

- Reversible gate
- Various reversible gates
  - CNOT-based gates
    - NOT, CNOT, C^2NOT (Toffoli), …
  - Generalized Toffoli gate
    - Positive controls
    - Negative controls





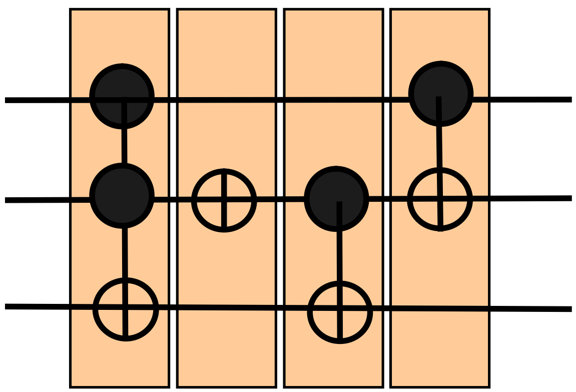




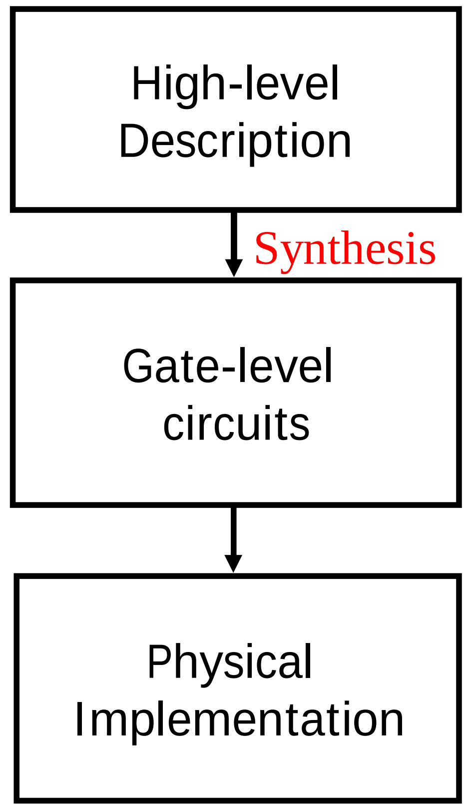
Matrix representation

- An n-qubit gate has a unitary $2^n \times 2^n$ matrix, QMatrix, describing its functionality.
- The QMatrix of an n-qubit quantum circuit is well-formed if it has the following two conditions:
  - Matrix elements can only be zeros or ones.
  - Each column or row has exactly one element with a value of 1.
- CNOT-based quantum circuits & Boolean reversible circuits have well-formed QMatrices

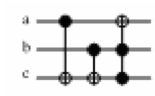
Reversible Circuits

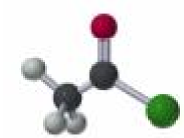


$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$



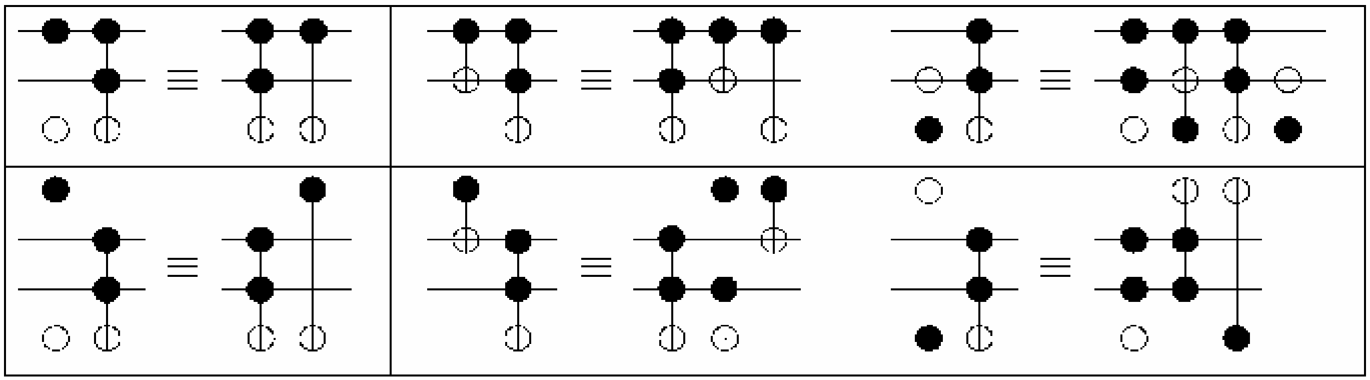
c_{in}	a	b	g_1	g_2	s
0	0	0	0	0	0
0	0	1	1	1	1
0	1	0	1	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	0	1	1





Synthesis Algorithms Categories

- Transformation-based algorithms
  - Used to improve the cost of circuit
  - Applied on the results of other algorithms
  - Usually use templates to optimize a circuit



Synthesis Algorithms Categories (Cnt'd)

- Constructive algorithms
  - Construct a circuit from a given specification (i.e. truth table, PPRM expansion, decision diagrams, …)
  - The resulted cost may not be optimized
  - The time complexity of the algorithm may be too high

The Proposed Algorithm

- Definition: L_k QTranslation
  - The application of a k-qubit gate with matrix G on a quantum circuit with a QMatrix M
  - The result of using an L_k QTranslation is the same as multiplication of M by G, i.e. MG
  - The result of using an L_k QTranslation is also well-formed

The Proposed Algorithm

- Definition: Quantum pair ($QPair_{i,j}$)
  - Two rows form a quantum matrix ($QPair_{i,j}$) if the numbers i and j differ in only one bit position

- Definition: C^kQPair
  - The 2^k rows of a QMatrix the row numbers of which have the same value on their n-k bit locations form a single group called C^kQPair

The Goal of the Algorithm

- The goal of MOSAIC is to decompose a given QMatrix into several elementary QMatrices of CNOT-based gates efficiently.
  - By generating a set of ordered L_k QTranslation
  - When applied to the QMatrix M, generates an identity matrix I

Applying an L_k QTranslation

- Lemma 1 and Lemma 2 explain the results of using an L_k QTranslation on a given QMatrix M

The MOSAIC Algorithm

Select the c^{th} column of the given QMatrix

set r to be the c row number which has a value of 1

if the r^{th} row is not marked as visited

if the b^{th} bits of r and c are not equal

find the number p which differs with r in its b^{th} bit

The MOSAIC Algorithm

set q to be the column number of row p which has a value of 1

if q != p and p >= r

exchange the locations of the p^{th} and r^{th} rows

mark the p^{th} and r^{th} rows as visited

Repeat the previous steps for all columns and all bits until M has been changed to identity matrix

Example (1)

c= 0 (00**0**)

b=0

p=6 (11**0**)

r=7 (111**1**)

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

q=7

- c: Brown box
- p: Green box

Example (2)

c= 1 (001)

r=0 (000)

p=1 (001)

q=2

$$
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

- Gray Box: visited rows

Example (3)

c= 2 (010)

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

r=1 (001)

visited

Example (4)

$$
\begin{array}{c c}
 & \begin{array}{cccccccc} & & & c=3\ (01\mathbf{1}) & & & & \end{array} \\
\begin{array}{r} \\ \\ r=0\ (01\mathbf{0}) \\ p=3\ (01\mathbf{1}) \\ \\ q=4 \\ \\ \\ \end{array} &
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

Example (5)

c= 4 (10**0**)

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

r=3 (01**1**)

visited

Example (6)

$$
\begin{array}{r c}
 & \begin{array}{cccccccc} & & & & & \text{c= 5 (101)} & & \end{array} \\
\begin{array}{r} \\ \\ \\ \\ \text{r=4 (100)} \\ \text{p=5 (101)} \\ \\ \\ \end{array} &
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

Example (7)

c= 6 (11**0**)

$$
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

r=5 (10**1**)

visited

Example (8)

c= 7 (11**1**)

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

r=6 (11**0**)
visited

Example (After the first step)

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Right locations

After the last step (identity matrix)

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Gate Extraction

- Each set of row exchanges corresponds to a gate.
- For example:
  - (6 & 7), (0 & 1), (2 & 3) and (4 & 5) swap operations correspond to a NOT gate applying on the last (b=0) qubit
  - (6 & 7), (2 & 3) swap operations correspond to a CNOT gate with the second qubit as its control and the last qubit as its target

The Algorithm Convergence

- Theorem 1: The MOSIC algorithm will converge to a possible implementation after several steps

The Time Complexity

- Assumption: At most *h* gates are needed
- Search-based method
  - $n \times 2^{n-1}$ gates must be evaluated to select the best possible gates at each step

$$C_n^1 + 2 \times C_n^2 + n \times (C_{n-1}^3 + \ldots + C_{n-1}^{n-1}) = n \times 2^{n-1}$$

  - $O(n \times 2^n)^h$ gates should be evaluated
- The MOSAIC algorithm needs $O(h \times 2^n)$ steps to reach a result

Experimental Results

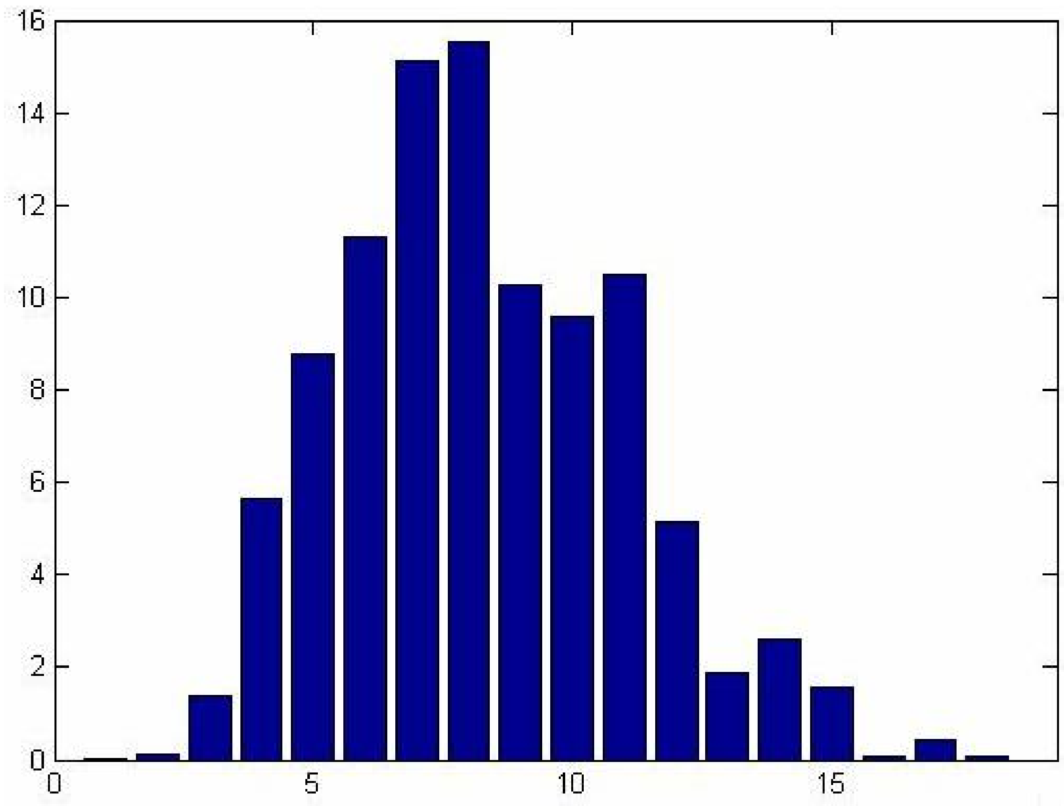
Ckt #	Specification	Number of Gates		Number of Searched Nodes & Steps		
		MOSAIC	[6],[7]	MOSAIC	[7]	[6]
1	(1,0,3,2,5,7,4,6)	4	4	40	15	11
2	(7,0,1,2,3,4,5,6)	3	3	24	300	761
3	(0,1,2,3,4,6,5,7)	3	3	32	10	7
4	(0,1,2,4,3,5,6,7)	7	5	64	786	156
5	(0,1,2,3,4,5,6,8,7,9,10,11,12,13,14,15)	9	7	160	8256	9515
6	(1,2,3,4,5,6,7,0)	3	3	24	4	4
7	(1,2,3,4,5,6,7,8,9,10,11,12,13,14, 15,0)	4	4	64	5	5
8	(0,7,6,9,4,11,10,13,8,15,14,1,12,3,2,5)	4	4	64	139	230

Experimental Results (Cnt'd)

	Specification	Number of Gates		Searched Nodes		
		MOSAIC	[6],[7]	MOSAIC	[7]	[6]
9	(3,6,2,5,7,1,0,4)	8	7	56	66	-
10	(1,2,7,5,6,3,0,4)	8	6	48	77	-
11	(4,3,0,2,7,5,6,1)	6	7	56	4387	-
12	(7,5,2,4,6,1,0,3)	6	7	32	352	-
13	(6,2,14,13,3,11,10,7,0,5,8,1,15,12,4,9)	19	15	192	678	-
14	(9,7,13,10,4,2,14,3,0,12,6,8,15,11,1,5)	23	14	240	9712	-
15	(6,4,11,0,9,8,12,2,15,5,3,7,10,13,14,1)	21	17	192	74521	-
16	(13,1,14,0,9,2,15,6,12,8,11,3,4,5,7,10)	29	16	352	85191	-
Average		**9.81**	**7.62**	**102**	**11531**	**-**

Experimental Results (Cnt'd)

- All possible 3-input/3-output reversible circuits (8!=40320) are synthesized



3-input/3-output reversible circuits

- Average number of gates per circuit
  - The proposed algorithm: 7.28
- Average number of steps per circuit = 63.87
- It takes about 4 minutes to synthesize all circuits
  - 0.006 seconds for each circuit on average

Different size QMatrices

Inputs	Number of Steps	Number of Gates	CPU Time (seconds)	Inputs	Number of Steps	Number of Gates	CPU Time (seconds)
1	1	1	0	2	7	2	0
3	34	4	0	4	155	9	0.01
5	624	17	0.05	6	2265	30	0.17
7	7731	55	0.51	8	24422	84	1.65
9	72960	133	5.46	10	225280	206	17.27
11	581632	259	45.39	12	1277952	312	61.50

Future Directions

- Working on the improvement of the resulting synthesized circuit
  - By combining the proposed approach and the search-based methods
  - By selecting the best possible variable at each step

Conclusions

- A new non-search based synthesis algorithm was proposed
- Several examples taken from the literature are used
- The proposed approach guarantees a result for any arbitrarily complex circuit
- It is much faster than the search-based ones

Thank you for your attention!