A Cycle-Based Synthesis Algorithm for Reversible Logic

Zahra Sasanian*, Mehdi Saeedi, Mehdi Sedighi, Morteza Saheb Zamani

{sasanian, msaeedi, msedighi, szamani}@aut.ac.ir

Quantum Design Automation Lab, Computer Engineering Department
Amirkabir University of Technology
Tehran, Iran

ASP-DAC 2009

Outline

- Introduction
- Basic Concepts
- Previous Work
- Synthesis Algorithm
- Experimental Results
- Future Work
- Conclusions

Introduction

- Reversible Logic
  - Equal number of inputs and outputs
  - Injective mapping
  - Example: f={0,1,3,5,2,6,7,4}

f: input → Output

0 → 0
1 → 1
2 → 3
3 → 5
4 → 2
5 → 6
6 → 7
7 → 4

Irreversible AND

i_1	i_2	i_3	f_1	f_2	f_3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	1	0	0

Power Dissipation

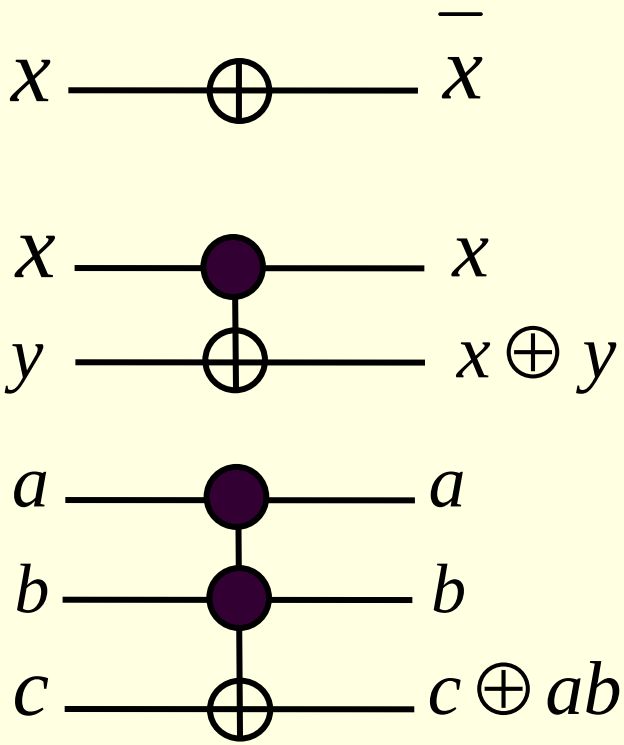
- **Rolf Landauer (1961)**
  - Every lost bit causes an energy loss
  - When a computer erases a bit of information, the amount of energy dissipated into the environment is at least $KT \times \ln 2$
- **Charles Bennett (1973)**
  - To avoid power dissipation in a circuit, the circuit must be built with reversible gates

Motivation

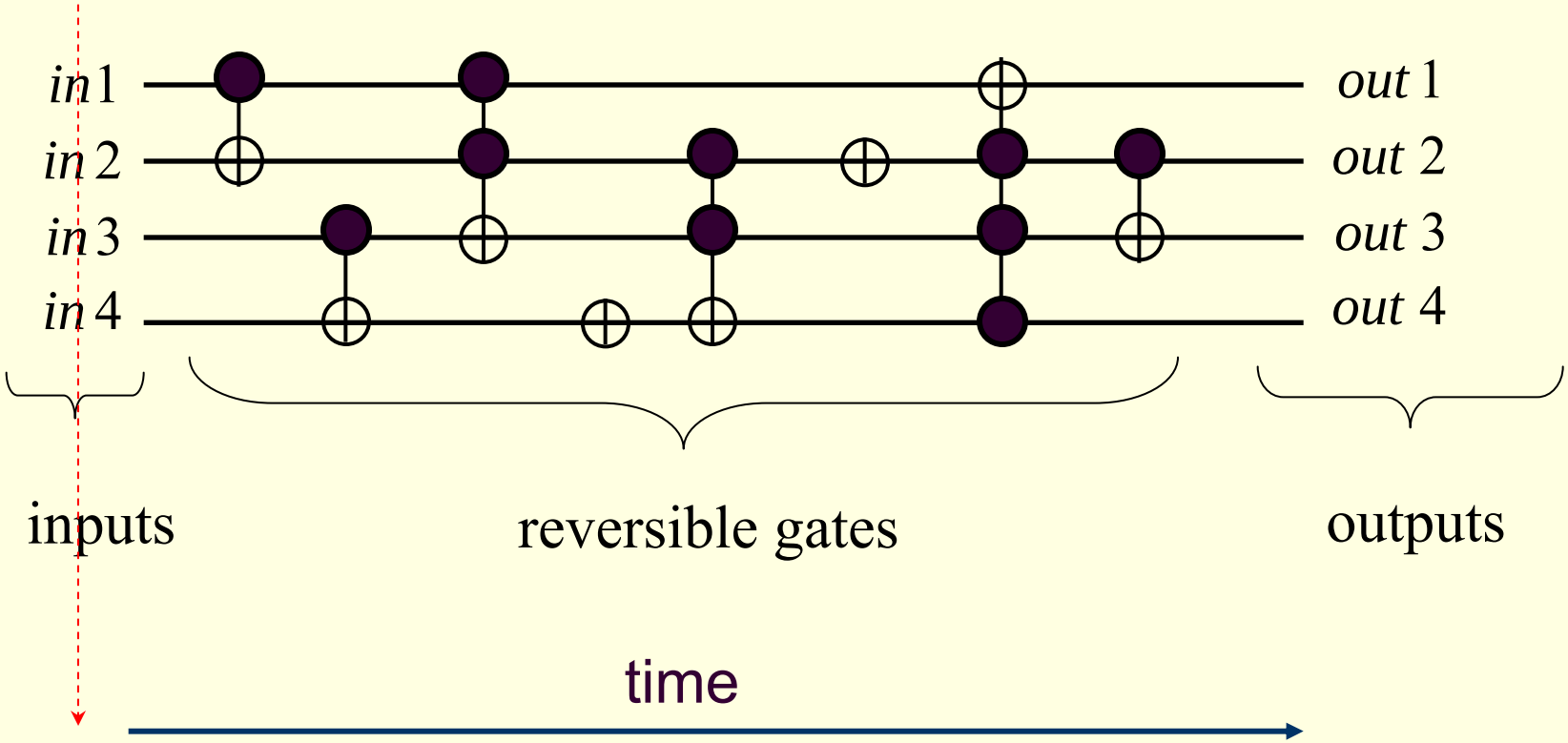
- Decrease in power dissipation
- Application in
  - Low power CMOS design
  - Quantum computing
    - Each unitary quantum gate is intrinsically reversible

Basic Concepts

- Reversible gates
  - NOT
  - CNOT
  - C^{2}NOT (Toffoli)
  - Generalized Toffoli gate



Reversible Circuits



Basic Concepts

- Transposition $f = (a, b)$
- K-Cycle $f = (a_1, a_2, \ldots, a_k)$

  $f(a_1)=a_2$, $f(a_2)=a_3$, …, $f(a_k)=a_1$
- Disjoint Cycles
  - Cycles f and g are called *disjoint* if they have no common members, i.e., $\forall\ a \in f,\ a \notin g$ and vice versa

Canonical Cycle Form (CCF)

- Every permutation function can be written uniquely, except for the order, as a product of disjoint cycles

$$f = (a_1, a_2, \ldots, a_k)(b_1, b_2, \ldots, b_j)(c_1, c_2, \ldots, c_j)$$

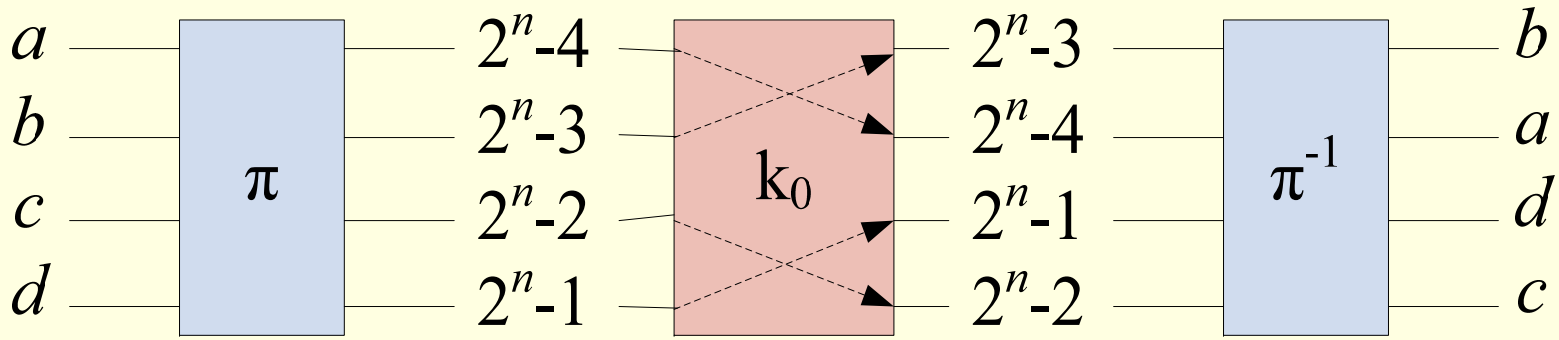
Previous Work

- Synthesis Algorithm of [6]
  - "Synthesis of Reversible Logic Circuits," *TCAD 2003*
  - Uses NCT (NOT, CNOT, Toffoli) library
  - Decomposes every cycle with length larger than two to a set of pairs of disjoint transpositions

$$(x_0,x_1,x_2,\ldots,x_k) = (x_0,x_1)(x_{k-1},x_k)(x_0,x_2,x_3,\ldots,x_{k-1})$$

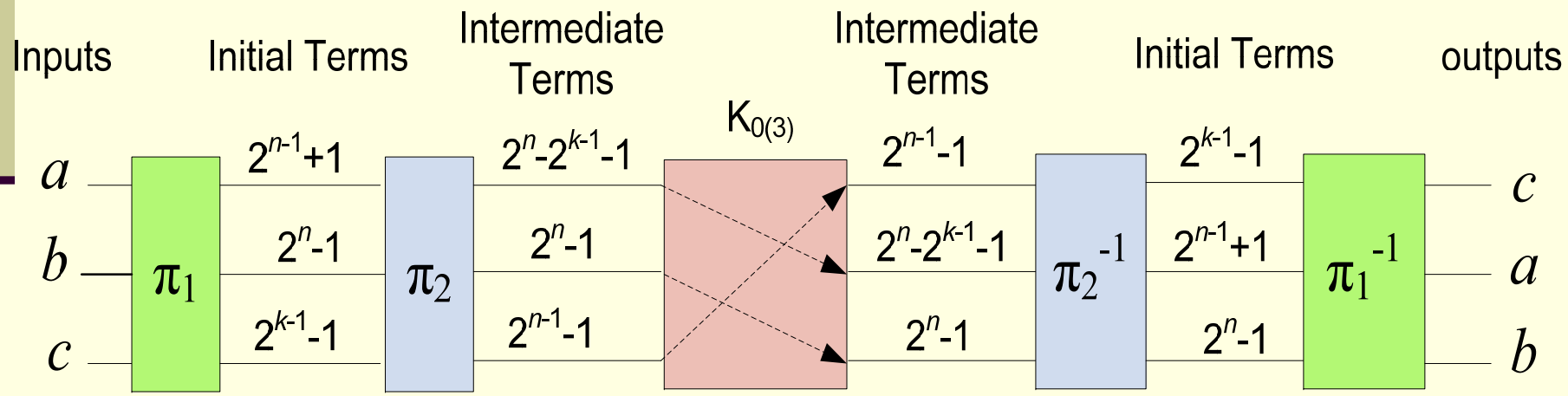
Previous Work

- Synthesis Algorithm of [6]
  - Synthesizes each disjoint transposition pair $(a, b)(c, d)$ using $\pi k_0 \pi^{-1}$ circuit



Cycle-Based Synthesis Algorithm

- Goal: To show the effect of synthesizing larger cycles directly
- Direct synthesis of 3-Cycles

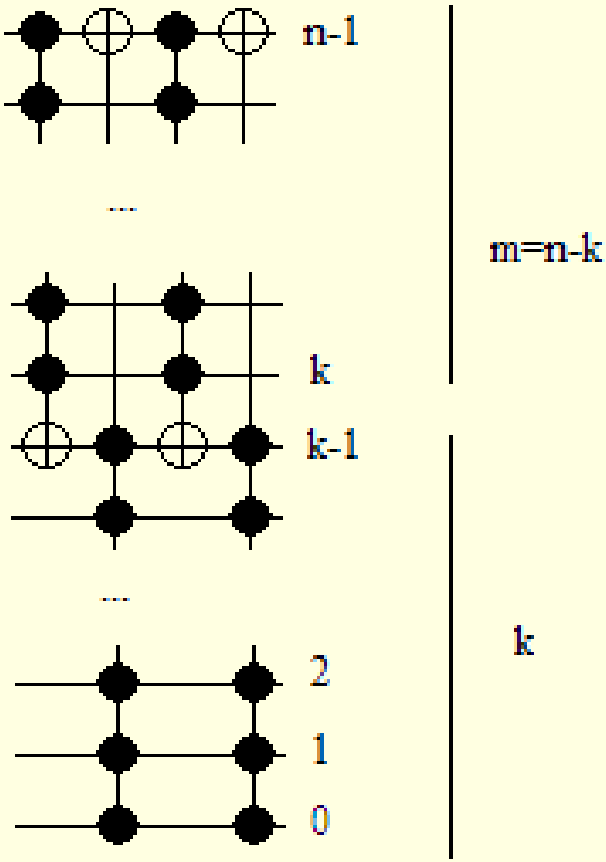


Cycle-Based Synthesis Algorithm

- 3-Cycle generator $k_{0(3)}$

$k_{0(3)}=(2^n-2^{k-1}-1,\ 2^n-1,\ 2^{n-1}-1)$

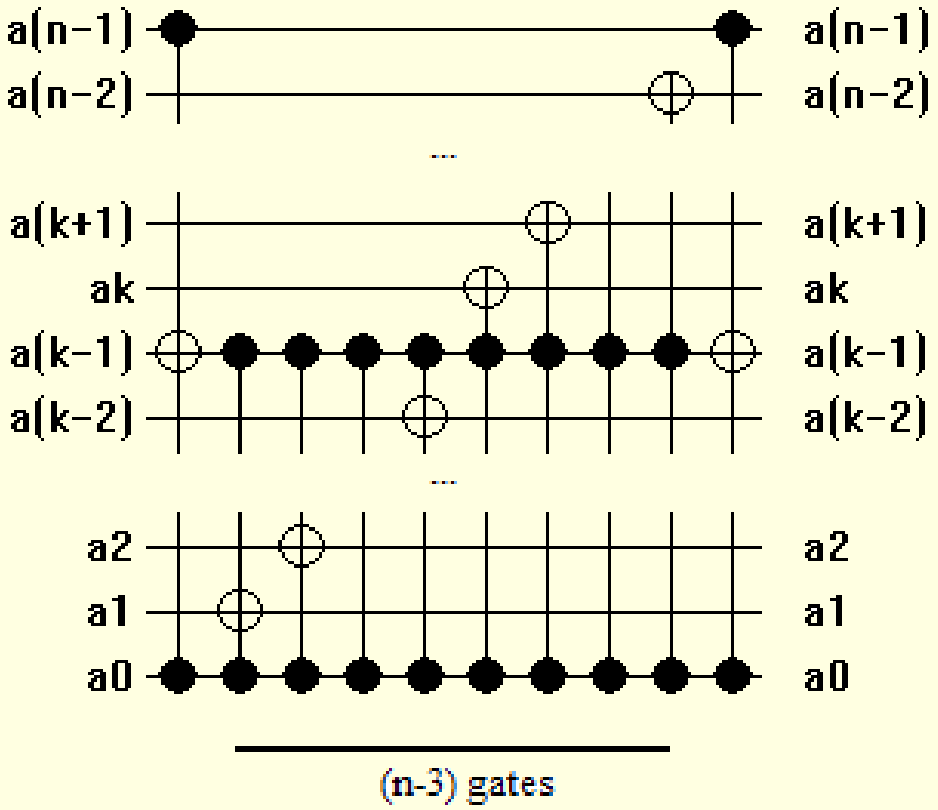
$k=\lceil n/2 \rceil$



Cycle-Based Synthesis Algorithm

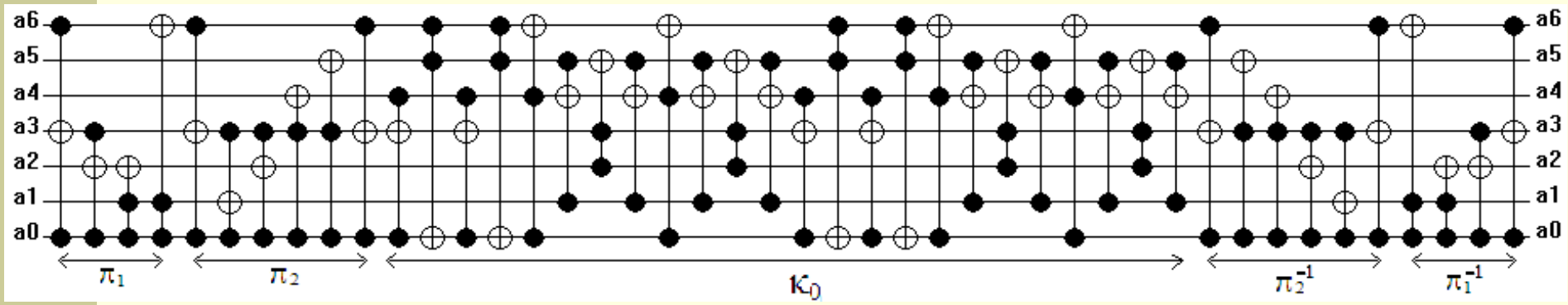
- π_2 circuit for every 3-cycle

$k=\lceil n/2 \rceil$



Example

- *f* = (73, 63, 13) , n=7
  - Using Toffoli gates



Building Blocks

Primitive Cycles	Initial terms	π_2 Circuit	Intermediate terms	K_0 Circuit ($k=\lceil n/2 \rceil$)	# of gates ([6]/ours)
(2-Cycle)(2-Cycle)	$(2^{n-1}+4, 2^{n-1}+1)(2^{n-1}+2, 2^{n-1}+7)$	T(0,n-1,2) T(1,n-1,2) T(2,n-1,[3, …, n-2])	$(2^n-4,2^n-3)(2^n-2,2^n-1)$	$C^{n-1}NOT(n-1,n-2,\ldots,2,0)$	18n-44/ 18n-44
(3-Cycle)	$(2^{n-1}-1,2^n-1,2^{k-1}-1)$	T(0,n-1,k-1) T(0,k-1,[1…k-2,k…n-2]) T(0,n-1,k-1)	$(2^n-2^{k-1}-1,2^n-1,2^{n-1}-1)$	$C^{n-k}NOT(n-1,\ldots,k,k-1)$ $C^kNOT(k-1,\ldots,0,n-1)$ $C^{n-k}NOT(n-1,\ldots,k,k-1)$ $C^kNOT(k-1,\ldots,0,n-1)$	36n-88/ 16n-34
(3-Cycle)(3-Cycle)	$(2^{k-1}-1,2^{n-1}-1,$ $2^{n-2}-1)$ $(2^{k-1}-2,2^{n-1}-2,$ $2^{n-2}-2)$	T(0, n-1,n-2) T(1, n-1,n-2) T(0,k-1,n-1) T(0,k-1,n-2) T(1,k-1,n-1) T(1,k-1,n-2) T(0, n-2,1) T(1, n-2,[2,...,n-3])	$(2^n-2^{k-1}-1,2^n-1,2^{n-1}-1)$ $(2^n-2^{k-1}-2,2^n-2,2^{n-1}-2)$	$C^{n-k}NOT(n-1,\ldots,k,k-1)$ $C^{k-1}NOT(k-1,\ldots,1,n-1)$ $C^{n-k}NOT(n-1,\ldots,k,k-1)$ $C^{k-1}NOT(k-1,\ldots,1,n-1)$	36n-88/ 22n-34

Partitioning

$f = (a, b, c, d, e, f, g, h, i, j, k, l, m)$

$= (a, b, c)\ (d, e, f)\ (g, h, i)\ (j, k, l)\ (m, a, d, g, j)$

$= (a, b, c)\ (d, e, f)\ (g, h, i)\ (j, k, l)\ (m, a, d)\ (g, j, m)$

Number of Gates (ours) = 2(22n-34)+2(16n-34) = 76n-136

Number of Gates ([6]) = 12(18n-44) = 108n-264

Experimental Results

Circuits	# of Inputs	Elapsed Time (ms)		Number of gates		Imp. (%)
		[6]	ours	[6]	ours	
1	9	8	7	18450	12544	32.0
2	8	4	6	7512	5368	28.5
3	10	9	9	42304	28538	32.5
4	11	24	23	55156	42880	22.3
5	9	10	8	12944	10242	20.9
6	15	16	15	18784	11914	36.6
7	16	32	30	33798	21018	37.8
8	16	19	20	48848	30258	38.1
9	19	21	15	44876	27728	38.2
10	23	15	7	32172	19452	39.5

Experimental Results (Cont.)

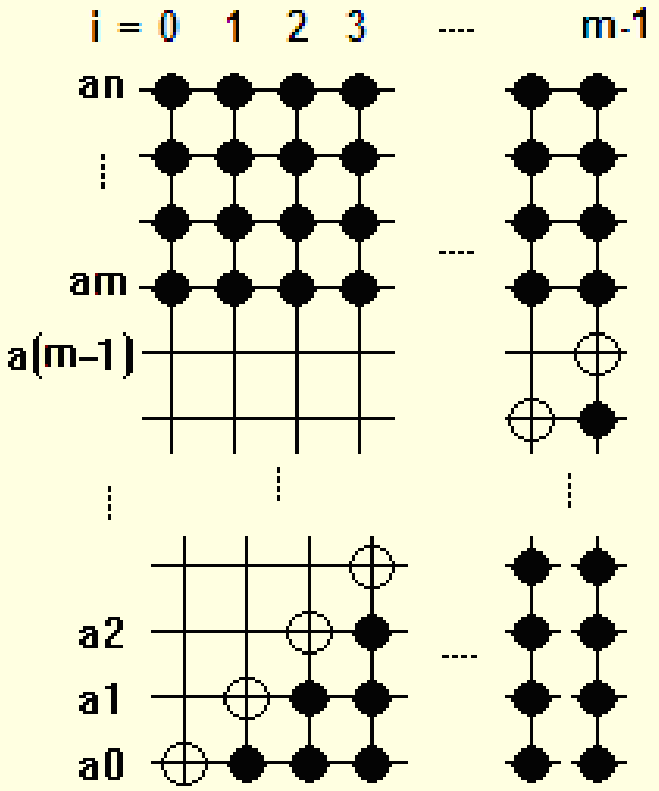
Circuits	# of Inputs	Elapsed Time (ms)		Number of gates		Imp. (%)
		[6]	ours	[6]	ours	
11	22	9	5	20036	12158	39.3
12	24	10	9	21800	13422	38.4
13	25	12	11	23646	14320	39.4
14	23	7	6	15468	9138	40.9
15	25	16	15	31184	18994	39.1
16	28	17	19	30568	18690	38.9
17	30	31	19	30308	17938	40.8
18	10	11	12	24728	19312	21.9
19	10	15	10	41452	27758	33.0
20	10	18	21	46224	31302	32.3
Average	**17.15**	**15.2**	**13.35**	**26600**	**19648.7**	**34.5**

Future Directions

- Generalization of the Cycle-Based Algorithm

  e.g. K_0 circuit for the 2^m-cycles

  - Is there an optimum cycle length?



Conclusions

- A new synthesis algorithm was proposed using direct synthesis of cycles
- The proposed algorithm uses simple NCT gates with no extra garbage bits
- The run time of the proposed synthesis algorithm is negligible
- The results show 34% improvement in number of generated gates with respect to [6]


Thanks