A Formulation of Fast Carry Chains Suitable for Efficient Implementation with Majority Elements

Behrooz Parhami **(2nd author)**

Dept. Electrical & Computer Eng.
Univ. of California, Santa Barbara

Ghassem Jaberipur

Shahid Beheshti Univ. & IPM, Iran

Dariush Abedi

Shahid Beheshti Univ., Iran

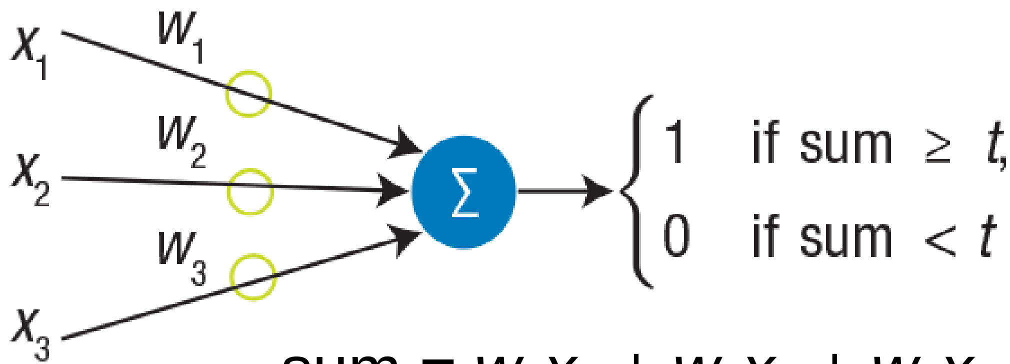
Continual Reassessment of Designs

- Change in cost/delay models with advent of ICs
  Transistors became faster/cheaper; wires costlier/slower
- Adaptation to CMOS, domino logic, and the like
  Optimal design for one technology not best with another
- Power and energy-efficiency considerations
  Voltage levels and number of transitions became important
- Quantum computing and reversible circuits
  Fan-out; managing constant inputs and garbage outputs
- Nanotech and process uncertainty / unreliability
  Designs for a wide range of circuit parameters and failures
- Novel circuit elements and design paradigms
  From designs optimized for FPGAs to biological computing

Threshold, Majority, Median

Threshold logic extensively studied since the 1940s

"Fires" if weighted sum of the inputs equals or exceeds the threshold value



sum = $w_1x_1 + w_2x_2 + w_3x_3$

Majority is a special case with unit weights and $t = (n + 1)/2$

For 3-input majority gate: $w_1 = w_2 = w_3 = 1$; $t = 2$

For 0-1 inputs, majority is the same as median

Axioms defining a median algebra

$$\mathrm{M}(a,b,b) = b$$

$$\mathrm{M}(a,b,c) = \mathrm{M}(a,c,b)\,; \qquad \mathrm{M}(a,b,c) = \mathrm{M}(c,a,b)$$

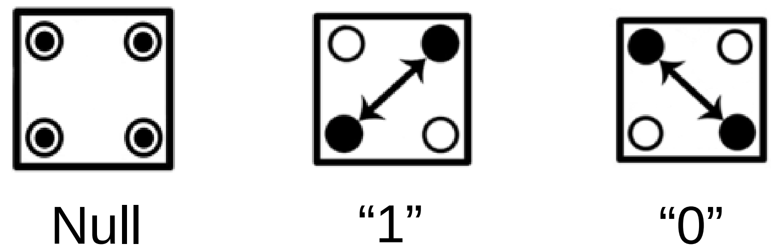
$$\mathrm{M}(\mathrm{M}(a,x,b),x,c) = \mathrm{M}(a,x,\mathrm{M}(b,x,c))$$

Emerging Majority-Based Technologies

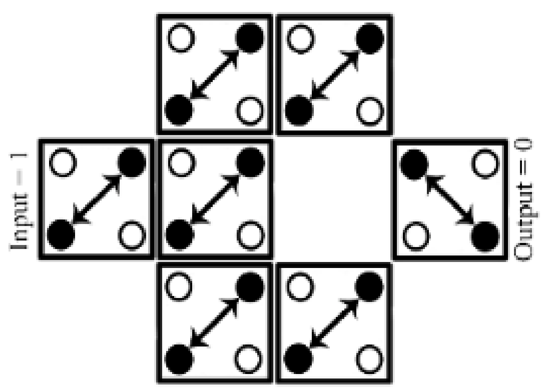
- Quantum-dot cellular automata (QCA)
  The basic cell has four electron place-holders (“dots”)
- Single-electron tunneling (SET)
  Based on controlled transfer of individual electrons
- Tunneling phase logic (TPL)
  Capacitively-coupled inputs feed a load capacitance
- Magnetic tunnel junction (MTJ)
  Uses two ferromagnetic thin-film layers, free and fixed
- Nano-scale bar magnets (NBM)
  Scaled-down adaptation of fairly old magnetic logic
- Biological embodiments of majority function
  Basis for neural computation in human / animal brains

Quantum-dot Cellular Automata (QCA)

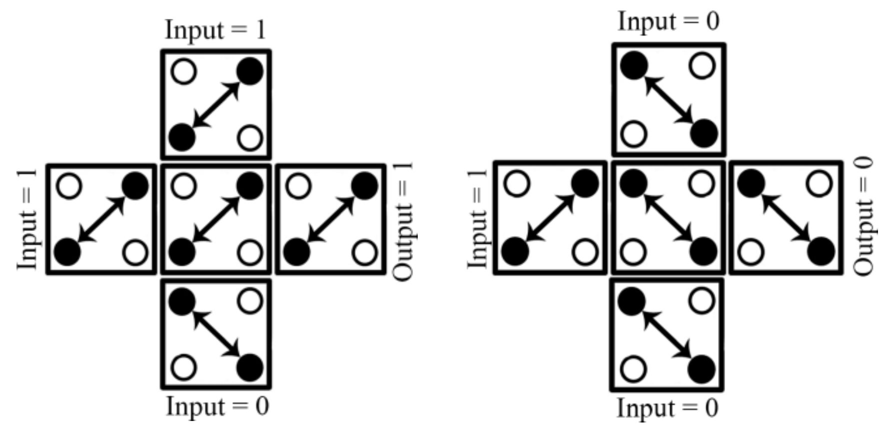
The basic cell has four electron place-holders ("dots")



Three QCA cell configurations



A robust QCA Inverter

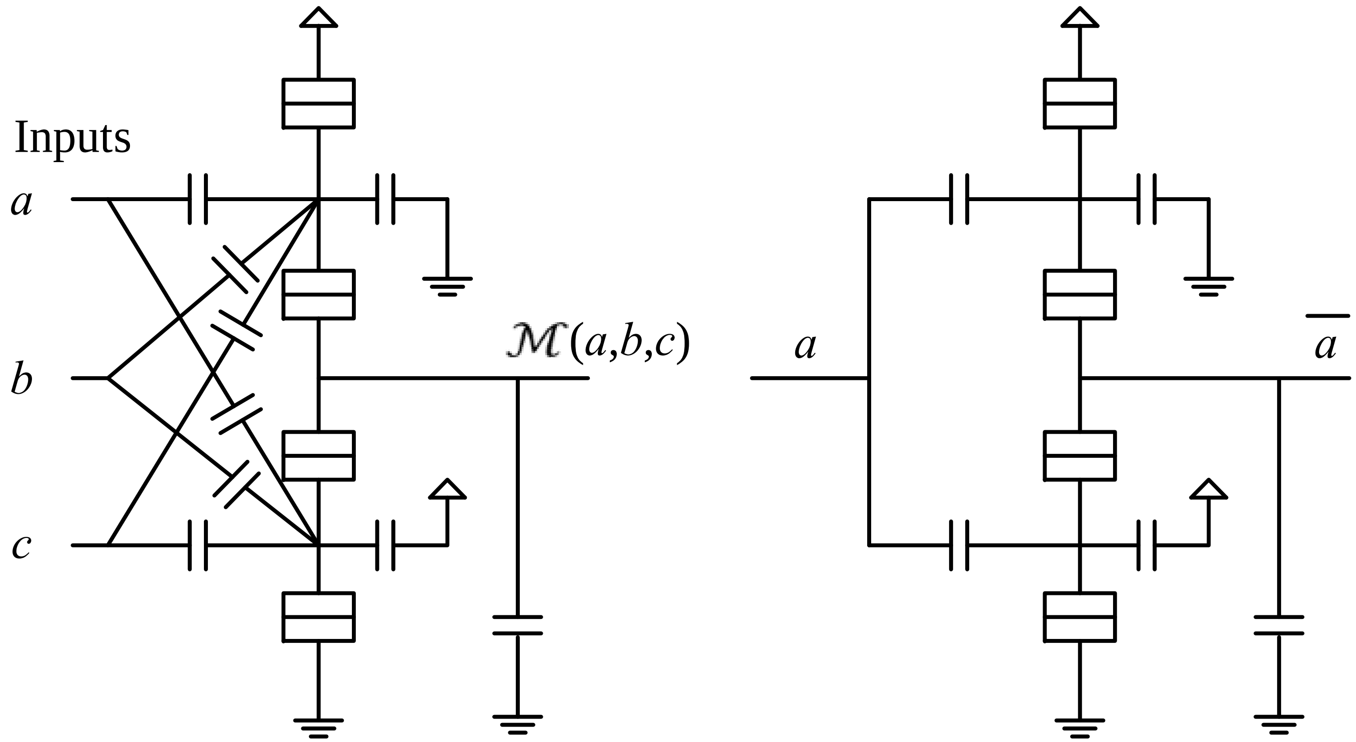


$M(1,1,0)=1$ $\qquad$ $M(0,1,0)=0$

QCA M gates with 2 sets of inputs

Single-Electron Tunneling (SET)

Based on controlled transfer of individual electrons



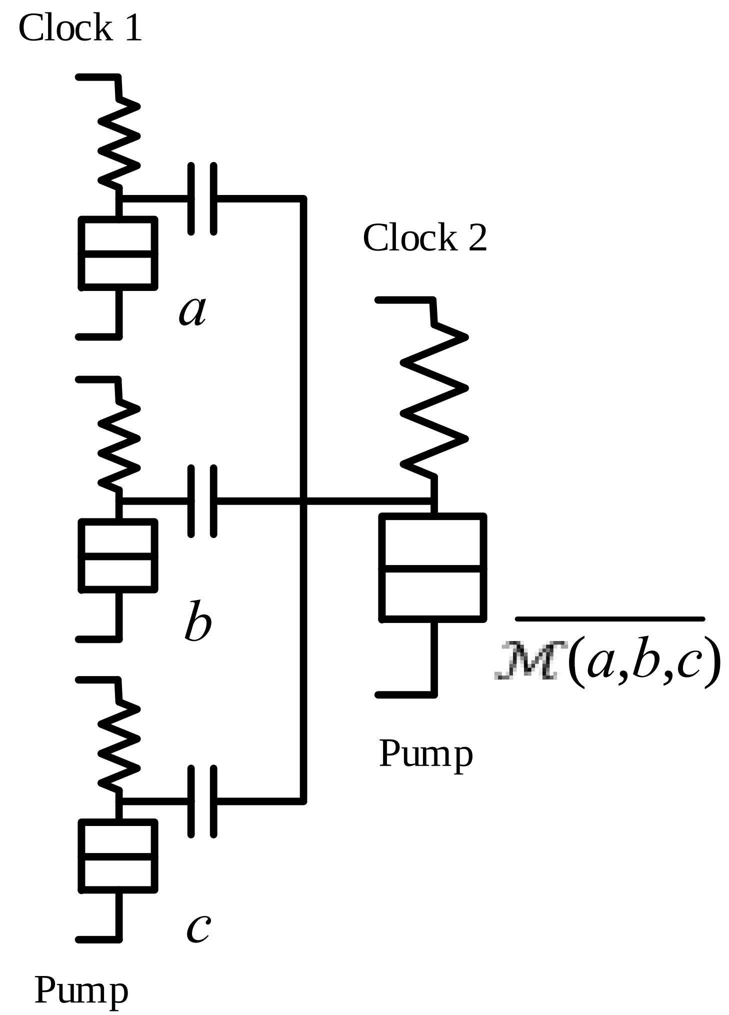
SET circuits for M (left) and inversion (right) [28]

Tunneling Phase Logic (TPL)

Capacitively-coupled inputs feed a load capacitance

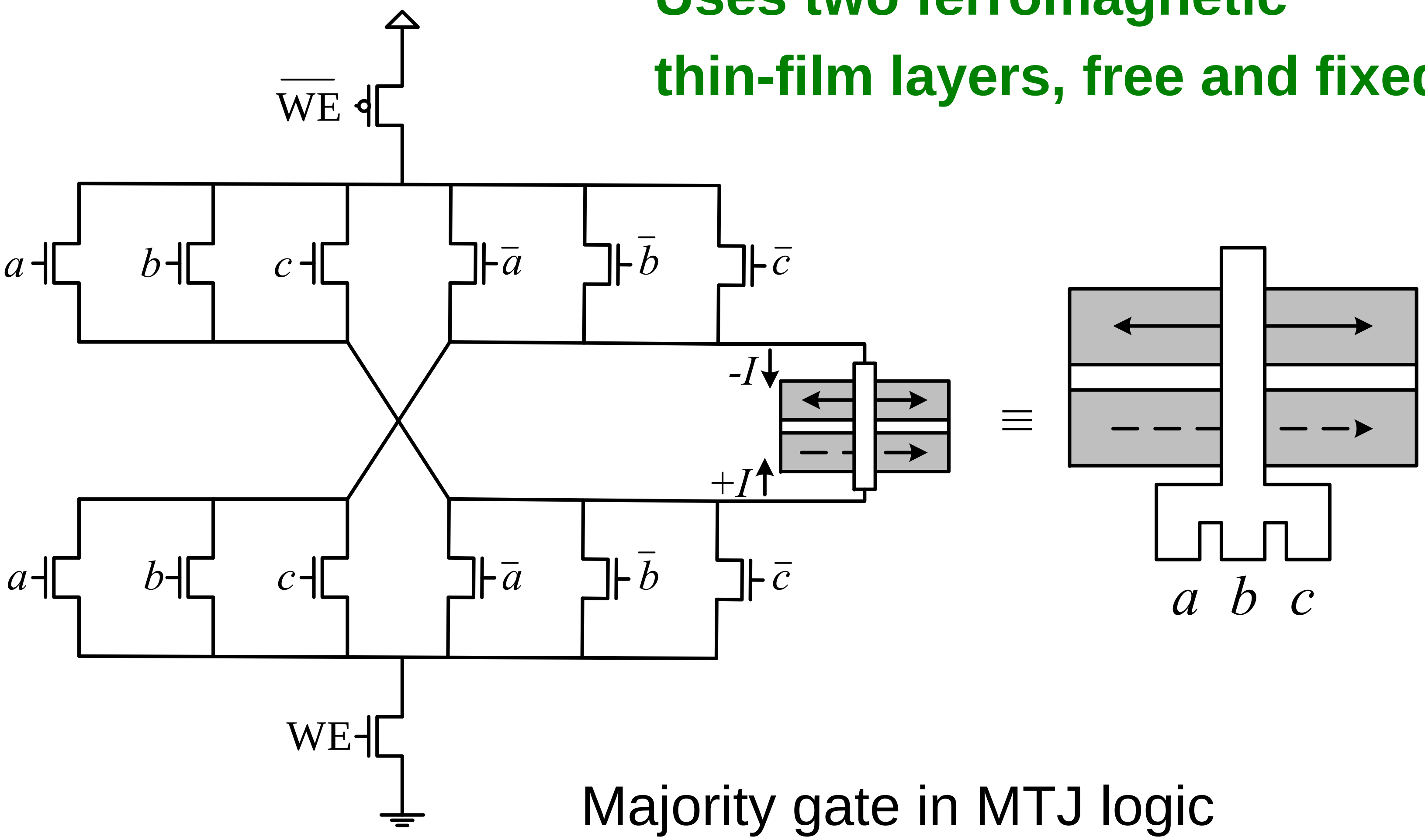
The basic TPL gate implements the minority function

inv(a) = $\overline{a}$ = minority(a, 0, 1)



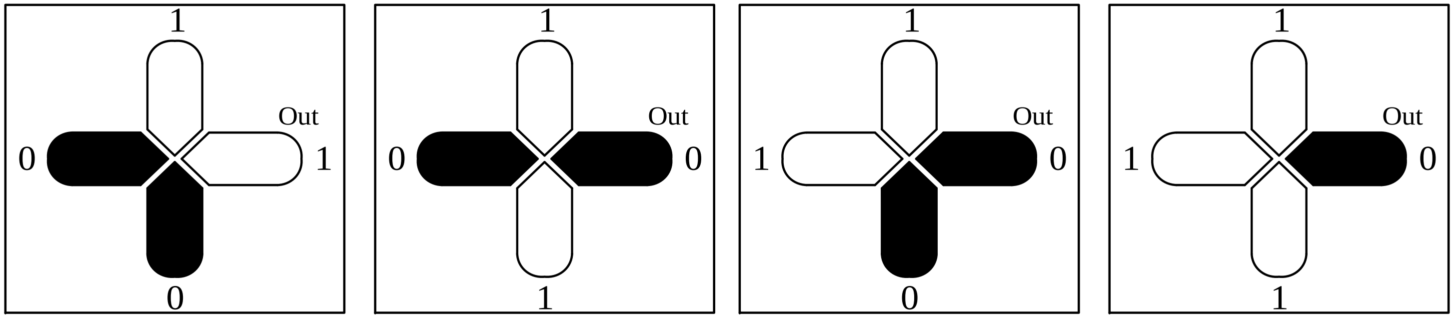
Magnetic Tunnel Junction (MTJ)

Uses two ferromagnetic thin-film layers, free and fixed



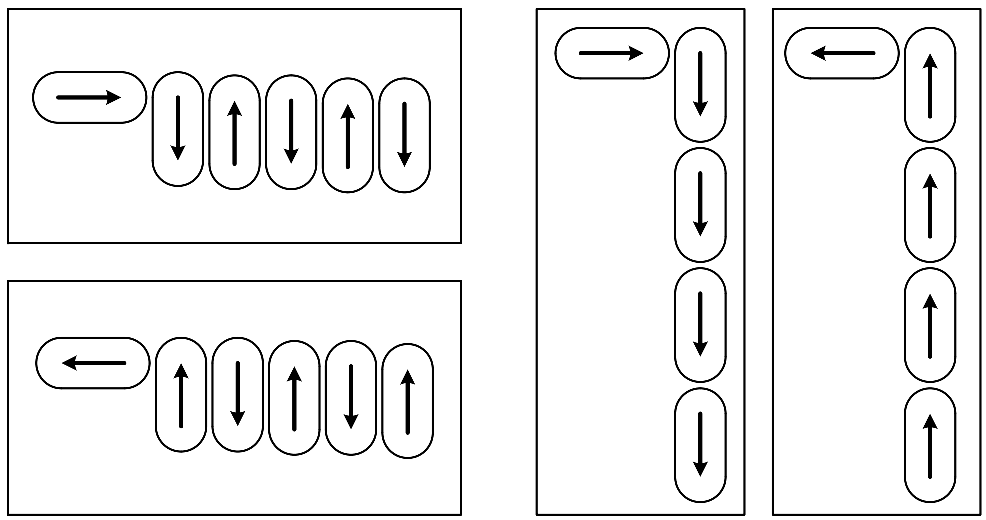
Majority gate in MTJ logic

Nano-scale Bar Magnets (NBM)



Voting with nanomagnets

Scaled-down adaptation of fairly old magnetic logic



Two types of nanomagnet wires

The Carry Recurrence and Operator

$$c_{i+1} = a_i b_i \vee (a_i \vee b_i) c_i \qquad 0 \le i \le n-1$$

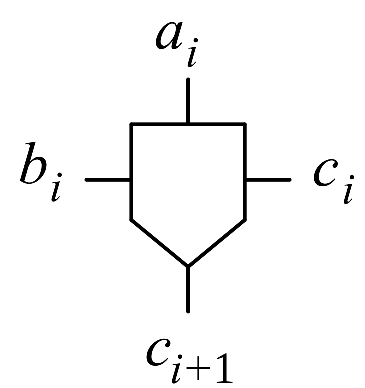
With *generate* $g_i = a_i b_i$ and *propagate* $p_i = a_i \vee b_i$ signals:

$$c_{i+1} = g_i \vee p_i c_i$$

With *group-generate* $G_{i:j}$ and *group-propagate* $P_{i:j}$ signals:

$$(G_{i:j}, P_{i:j}) = (G_{i:k} \vee P_{i:k} G_{k-1:j}, P_{i:k} P_{k-1:j})$$

$$c_{i+1} = G_{i:j} \vee P_{i:j} c_j$$



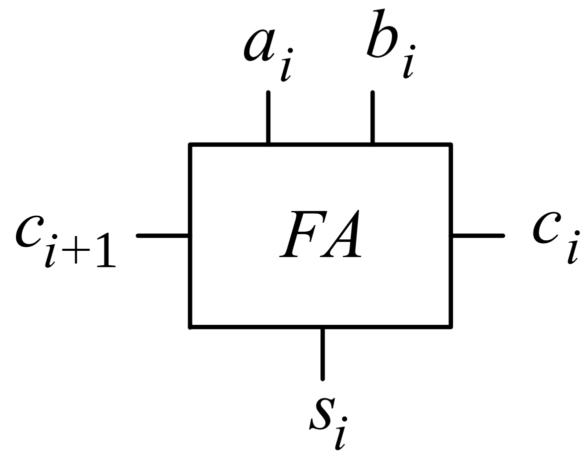
Carry generation using a majority gate:

$$c_{i+1} = M(a_i, b_i, c_i)$$

The Full-Adder (FA) Building Block

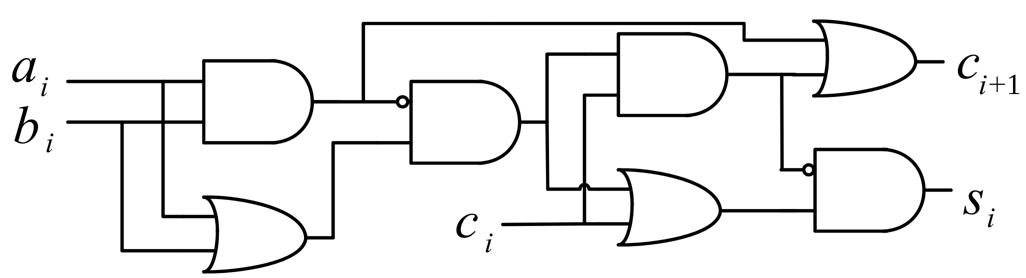
$$s_i = a_i \oplus b_i \oplus c_i$$

$$c_{i+1} = a_i b_i + (a_i + b_i)c_i$$



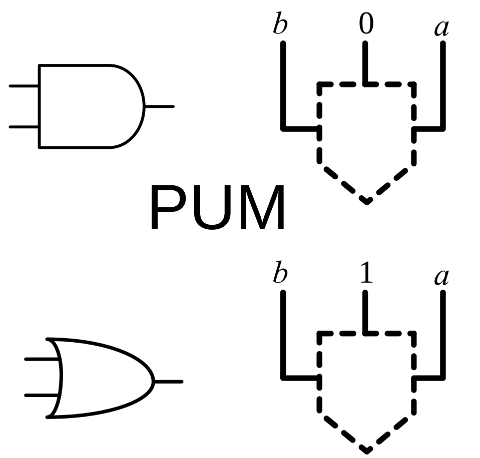
FA has been widely studied and optimized

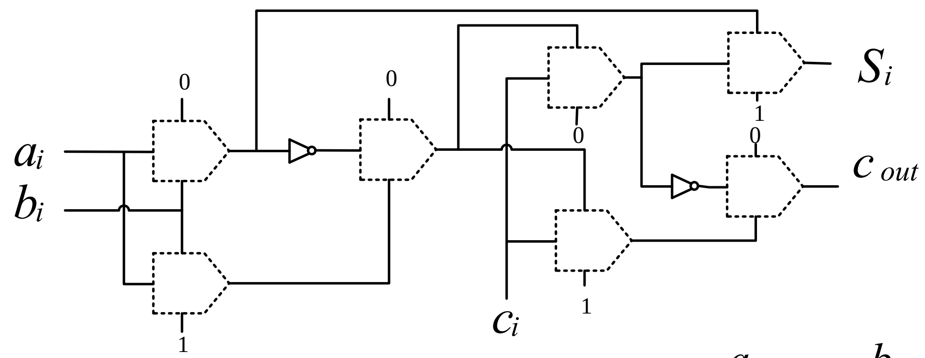
Implementation with seven 2-input gates:

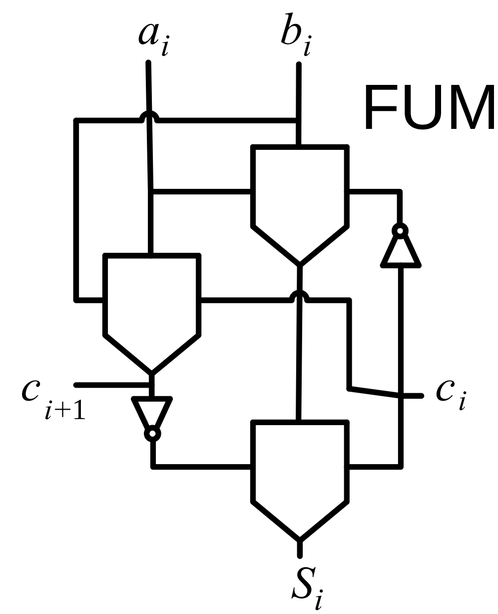


Majority-Gate Implementations of FA

Blind mapping: Seven partially utilized M-gates, 2 inverters:





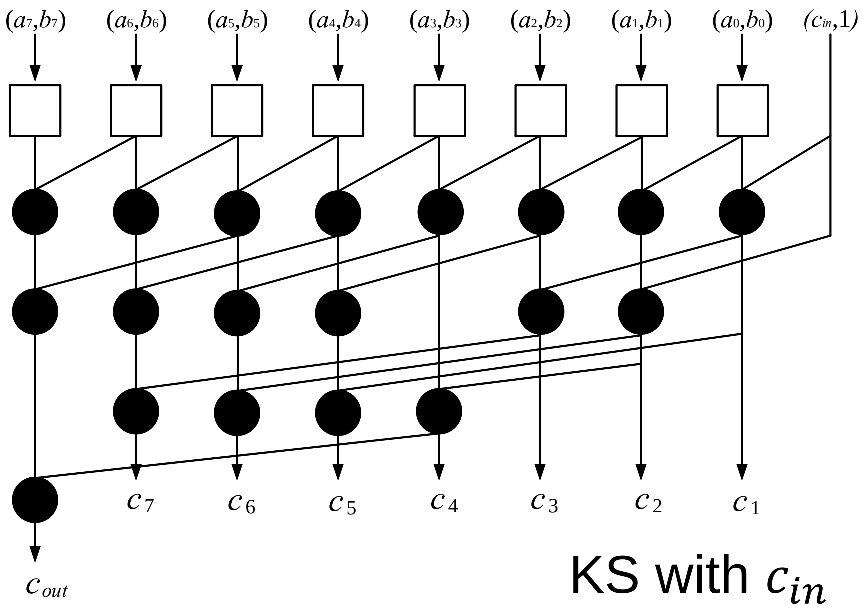


Three fully-utilized M gates, 2 inverters:

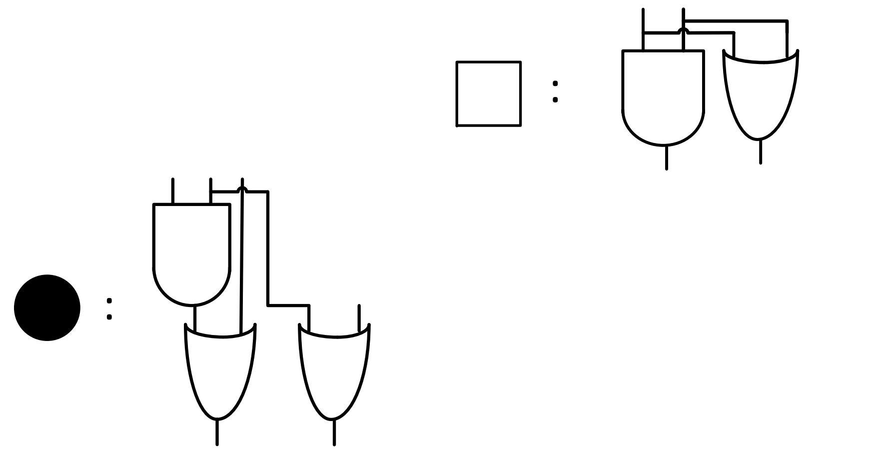
$$s_i = M(M(\overline{c}_i, a_i, b_i), \overline{M(a_i, b_i, c_i)}, c_i)$$

$$c_{i+1} = M(a_i, b_i, c_i)$$

Parallel-Prefix Kogge-Stone-Like CGN



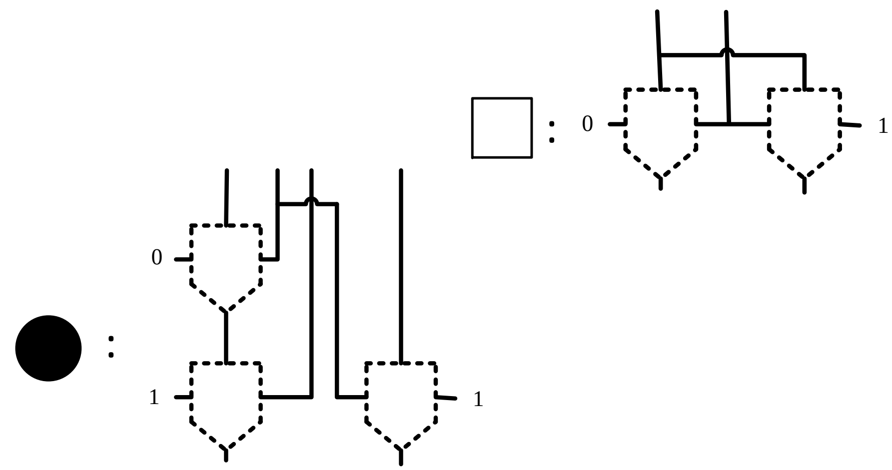
KS with c_{in}



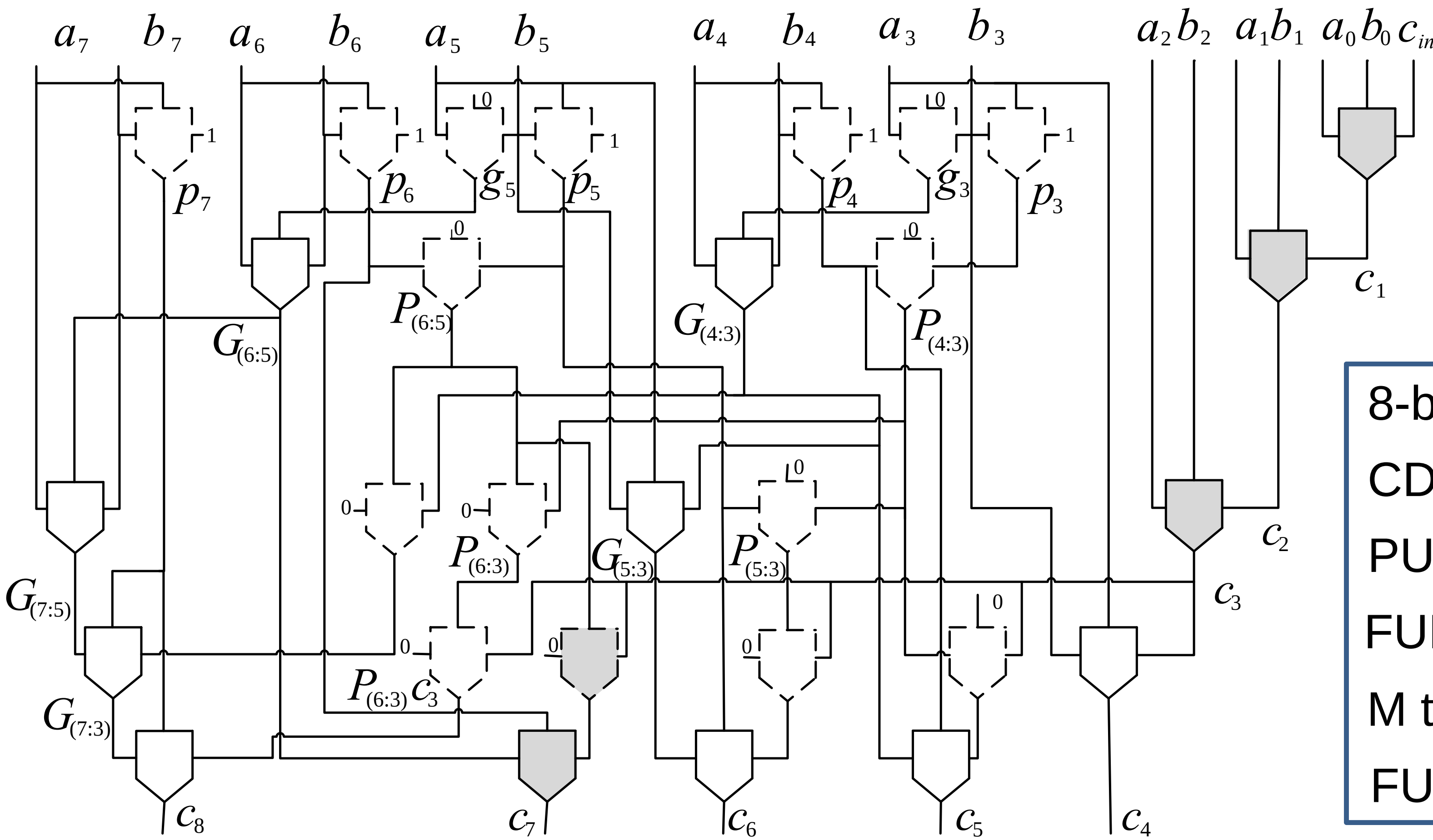
M-based implementations of the building blocks:

Blind mapping

Total of 73 PUM gates



Exploiting Fully Utilized M-Gates:
First Attempt by Pudi *et al.*



8-bit CGN:

CDP: 5 M

PUMs: 15

FUMs: 13

M total: 28

FUM%: 53

[61% fewer M-gates than with blind mapping]

Exploiting Fully Utilized M-Gates:
Second Attempt by Perri *et al.*

Two-bit CGN
with 1 M CDP
in c_i-to-c_{i+2} path

Total for 8-bit adder: 24

[67% fewer M-gates
than with blind mapping]

2-bit CGN:

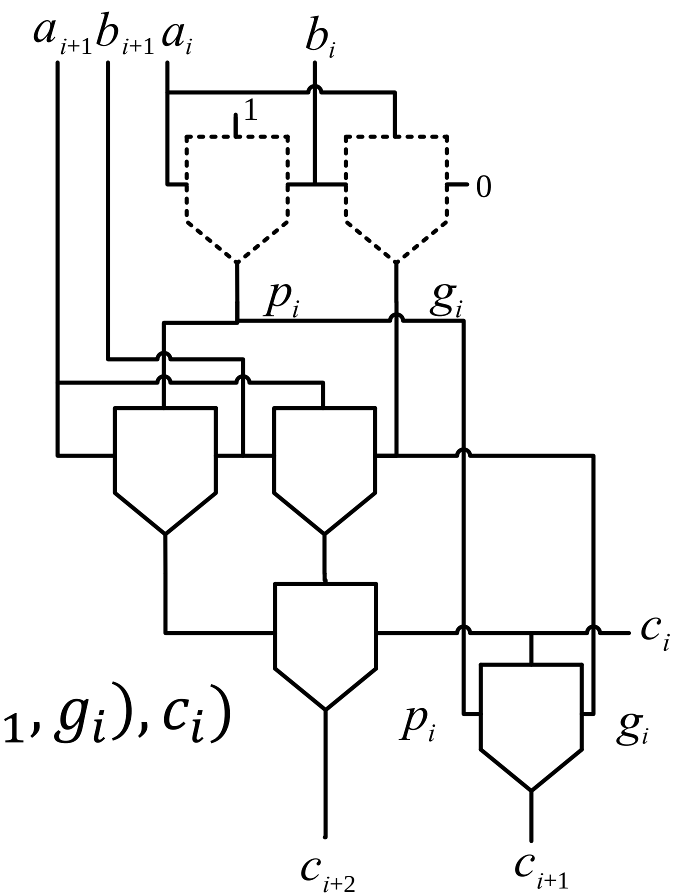
CDP: 1 M

PUMs: 2

FUMs: 4

M total: 6

FUM%: 67



$$c_{i+2} = M(M(a_{i+1}, b_{i+1}, p_i), M(a_{i+1}, b_{i+1}, g_i), c_i)$$

Conventional (2M delay, 2 FUM):

$$c_{i+2} = M(a_{i+1}, b_{i+1}, M(a_i, b_i, c_i))$$

Our Compromise Solution
(1M carry-path delay, 3 FUM)

$$c_{i+2} = M(M(a_{i+1}, b_{i+1}, a_i), M(a_{i+1}, b_{i+1}, b_i), c_i)$$

$$A_{i+1:i} = M(a_{i+1}, b_{i+1}, a_i)$$

$$B_{i+1:i} = M(a_{i+1}, b_{i+1}, b_i)$$

$$c_{i+2} = M(A_{i+1:i}, B_{i+1:i}, c_i)$$

Think of $A_{i+1:i}$ and $B_{i+1:i}$, as representing 2-bit inputs $a_{i+1}a_i$ and $b_{i+1}b_i$

Example:

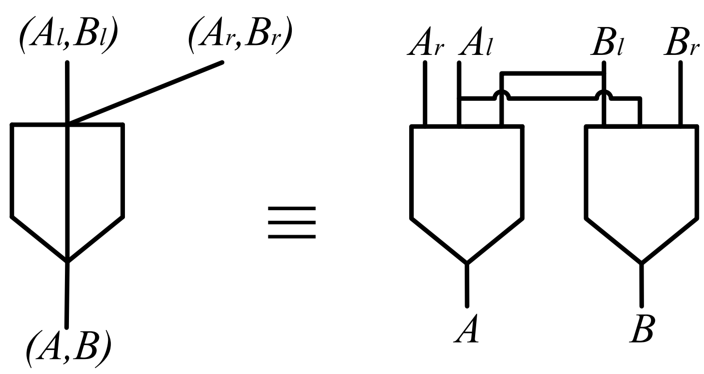
$$a_{i+1}a_i = c_i = 1 \Longrightarrow a_i = c_i = 1 \Longrightarrow$$

$c_{i+1} = 1$ and $a_{i+1} = 1 \Longrightarrow c_{i+2} = 1$

Generalizing the Compromise Solution

Twin M-gate:

$(A_{j:i}, B_{j:i})$: $(M(a_j, b_j, A_{j-1:i}), M(a_j, b_j, B_{j-1:i}))$



(A, B): $(M(A_l, B_l, A_r), M(A_l, B_l, B_r))$

Majority group generate and propagate:

$\Gamma_{j:i} = A_{j:i}B_{j:i}$ $\quad\quad \Pi_{j:i} = A_{j:i} + B_{j:i}$

$\Gamma_{j:i} = g_j + p_j\Gamma_{j-1:i}$ $\quad\quad \Pi_{j:i} = g_j + p_j\Pi_{j-1:i}$

Properties:

$$\Gamma_{j:i} = g_j + p_j\Gamma_{j-1:i}$$

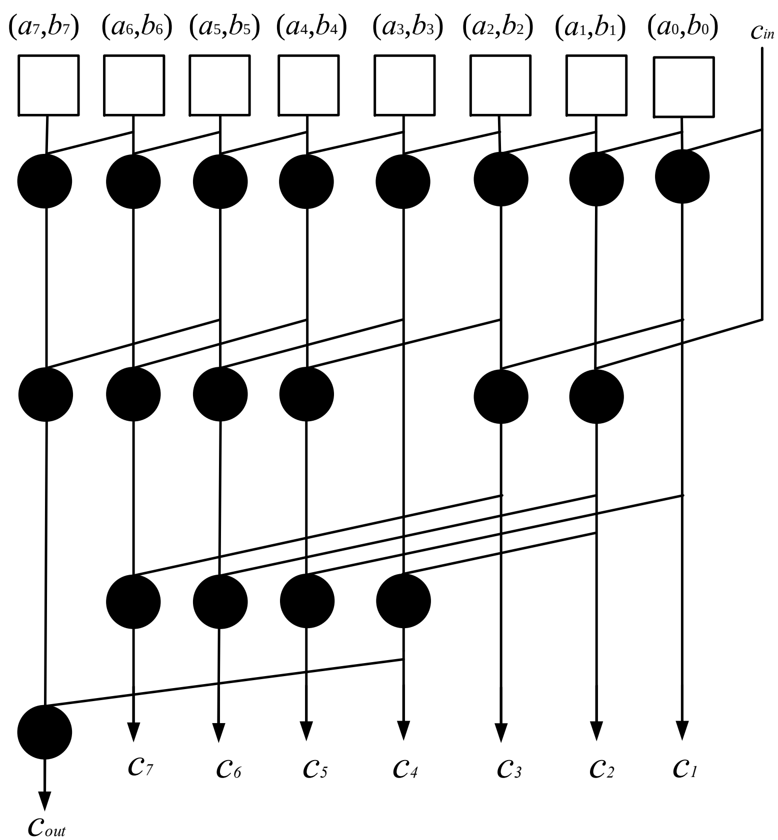
$$\Pi_{j:i} = g_j + p_j\Pi_{j-1:i}$$

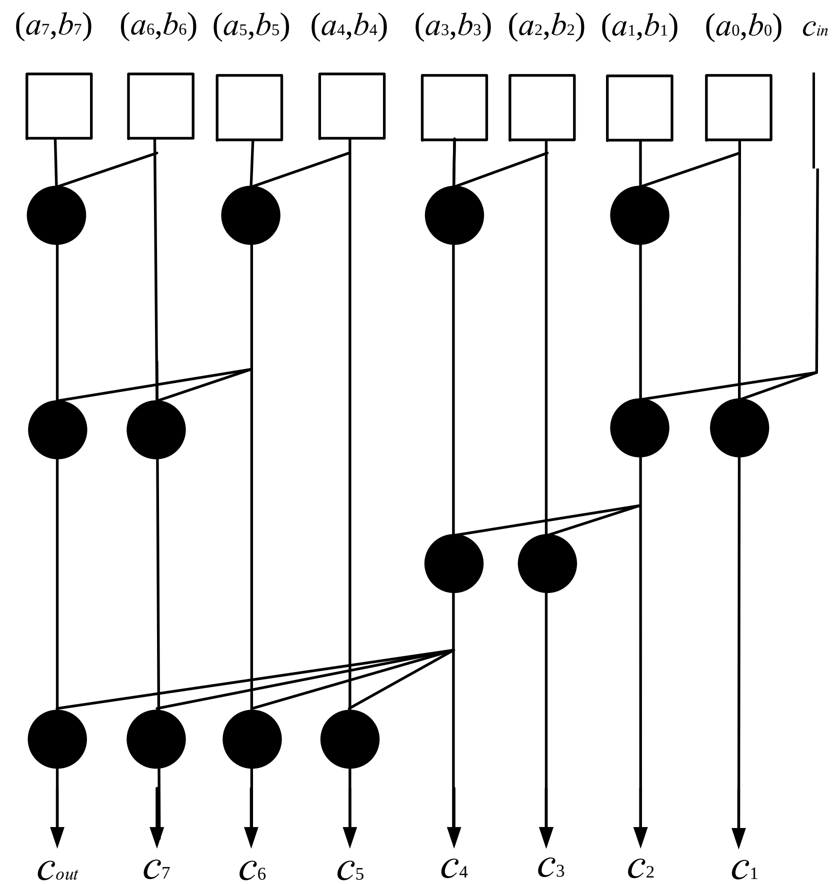
$$c_{i+j+1} = M(A_{i+j:i}, B_{i+j:i}, c_i)$$

Associativity:

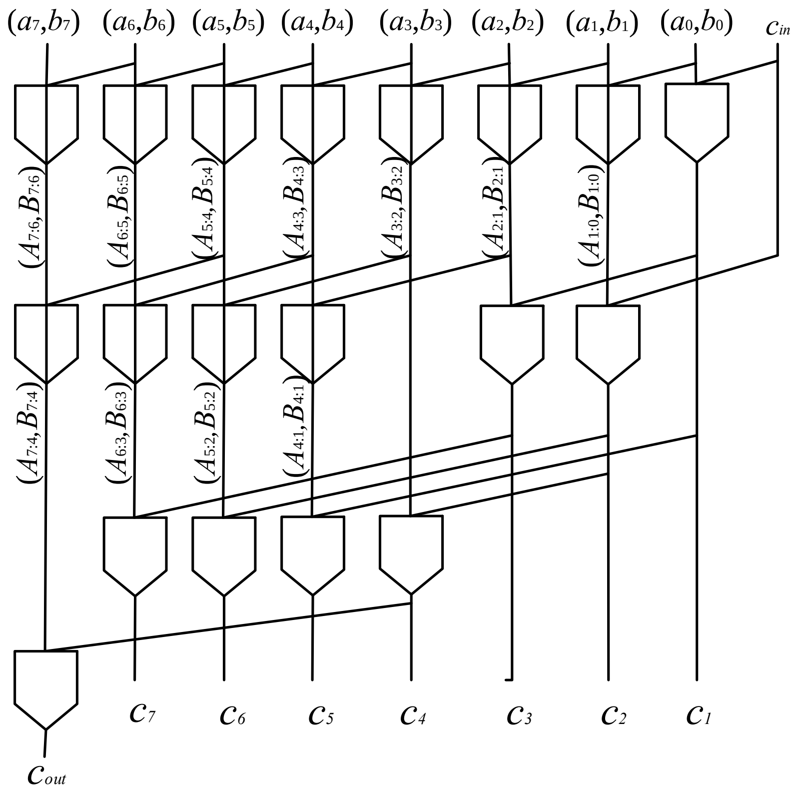
$$A_{k+j:i} = M(A_{k+j:j}, B_{k+j:j}, A_{j-1:i}),\ B_{k+j:i} = M(A_{k+j:j}, B_{k+j:j}, B_{j-1:i})$$

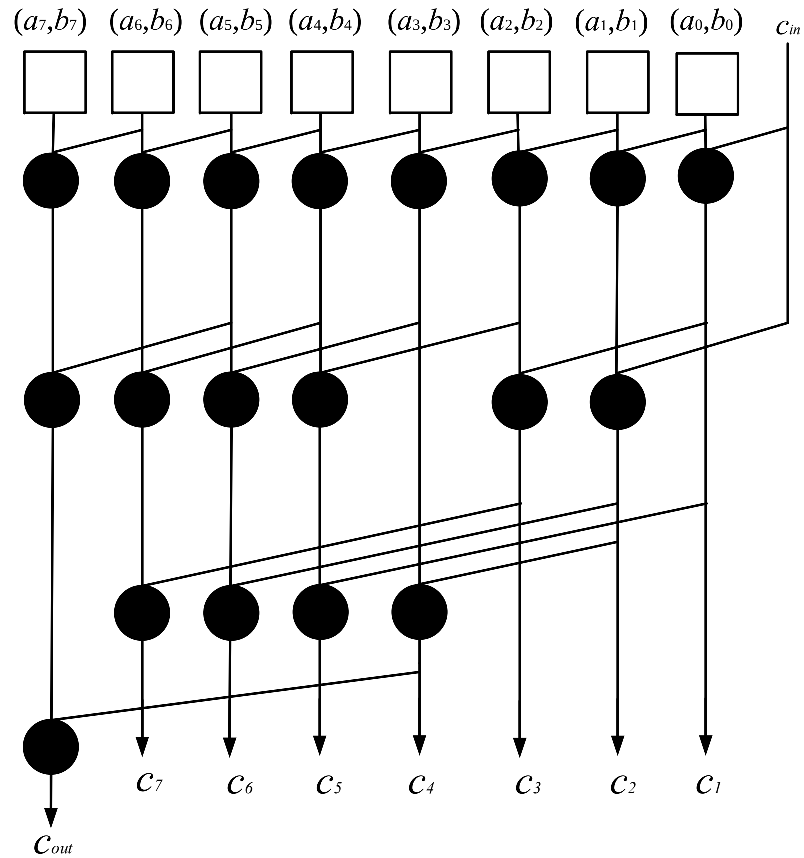
KS-Like and LF-Like M-Based CGNs
(with C_{in})



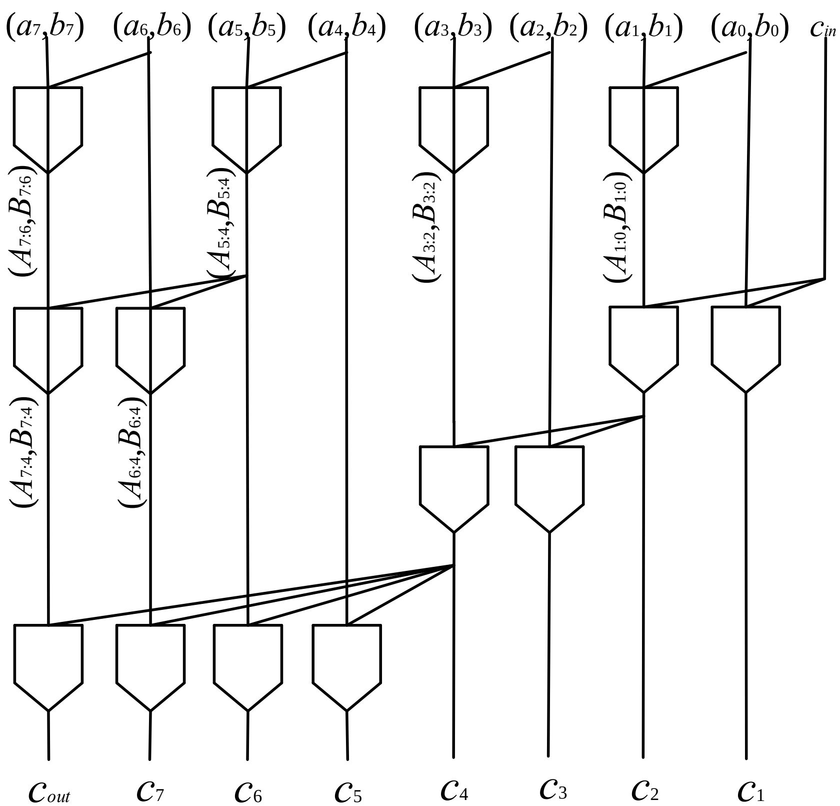


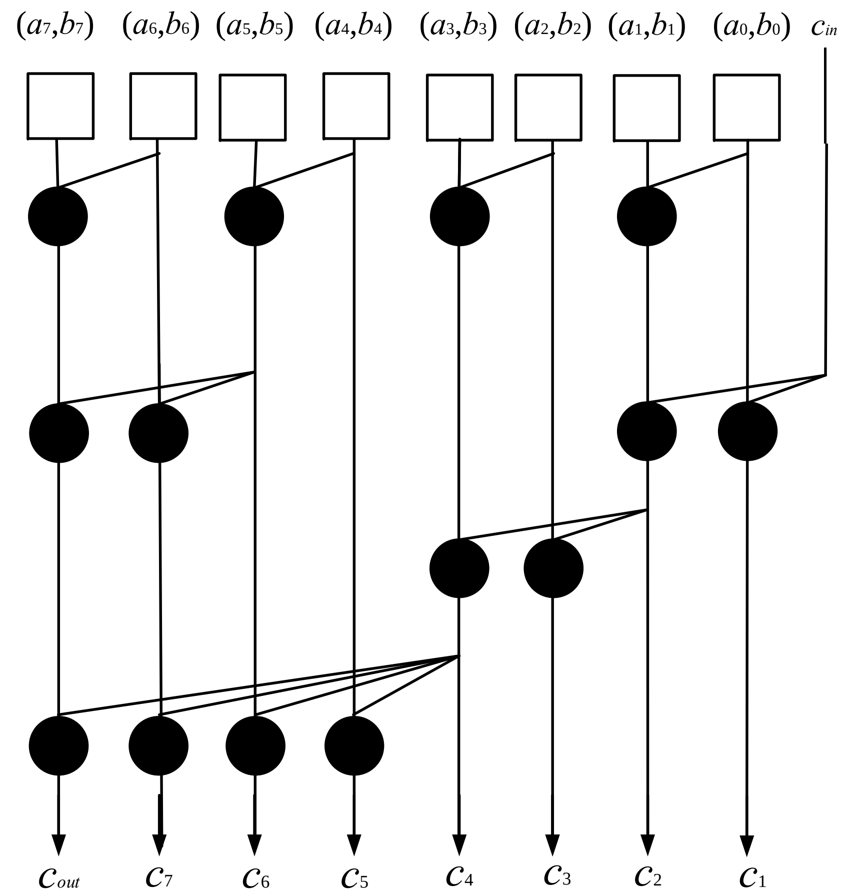
KS-Like M-Based CGNs
(with C_{in}) (% of FUM: 100)



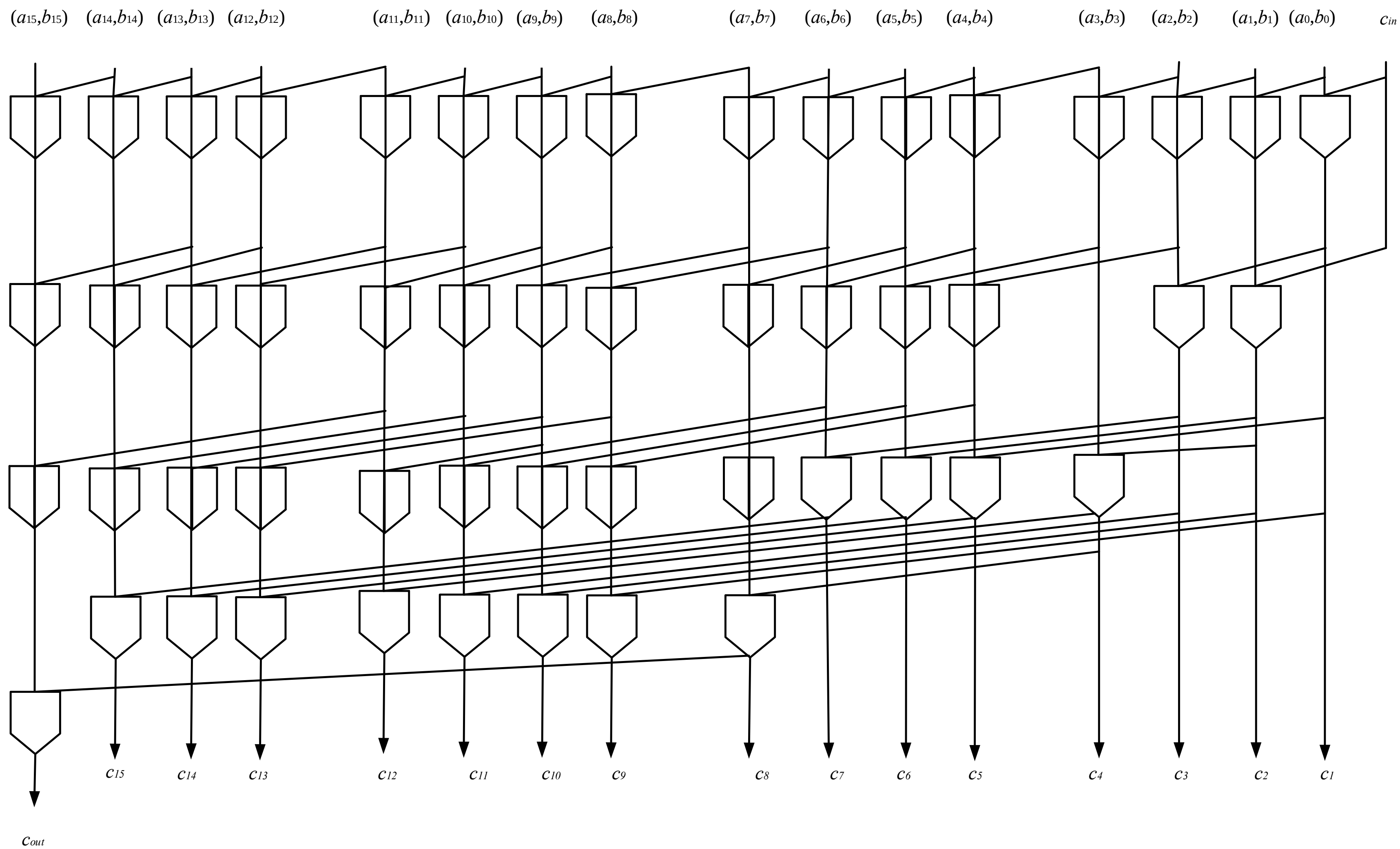


LF-Like M-Based CGNs
(with C_{in}) (% of FUM: 100)

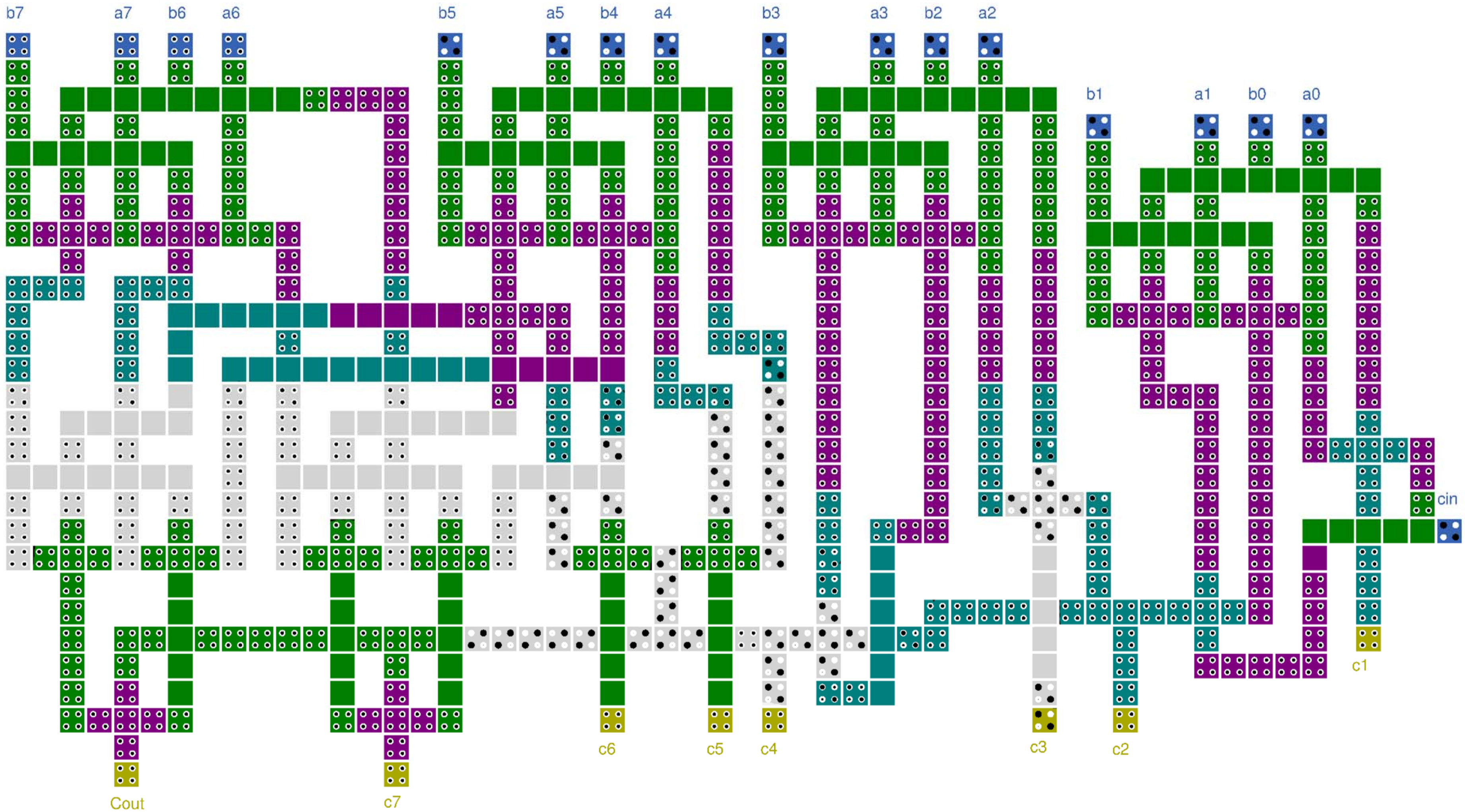




Scaling up to 16-bit KS-Like Design



QCA Implementation: 8-Bit LF-Like



Comparison with Previous Work
(8-bit CGN)

	Delay (clock zone)	PUM*	FUM*	Total M
New KS-like	6	0	30	30
New LF-like	6	0	20	20
[13]	9	28	7	35
[15]	9	15	13	28

* Partially / Fully-Utilized M-Gates

Conclusions and Future Work

- Best M-based carry-network designs to date
  - More efficient use of (fully utilized) M-gates
  - Applicable to a variety of PPN design styles
  - Benefits over naïve designs and prior attempts
- Majority-friendly tech's becoming important
  - Improve, assess, and fine-tune implementations
  - Extend designs to several other word widths
  - Obtain generalized cost / latency formulas
  - Pursue design methods for other technologies

Questions or Comments?

parhami@ece.ucsb.edu
http://www.ece.ucsb.edu/~parhami/
jaberipur@sbu.ac.ir