Majority Logic Synthesis: From CMOS To Emerging Technologies

Luca Amarù

Synopsys Inc., Mountain View, California, USA.

Outline

- Majority Logic Synthesis:
  - Why Majority Logic?
  - Majority Inverter Graph (MIG)
  - MIG Optimization
- MIG for Super Conducting Electronics (SCE):
  - SCE Brief Intro
  - Synthesis Challenges for SCE
  - MIG Optimization for SCE
- Conclusions

Acknowledgments

- The material of this presentation has been made possible thanks to several collaborations on MIG synthesis.
- MIG data structure and optimization was originally developed together with Pierre-Emmanuel Gaillardon and Giovanni De Micheli.
- Initial study on MIG application to SCE was done in collaboration with Jamil Kawa, Arturo Salz and Antun Domic.
- Further studies on MIG theory & applications was done in collaboration with Mathias Soeken, Eleonora Testa, Winston Haaswijk, Heinz Riener and Odysseas Zografos.

Outline

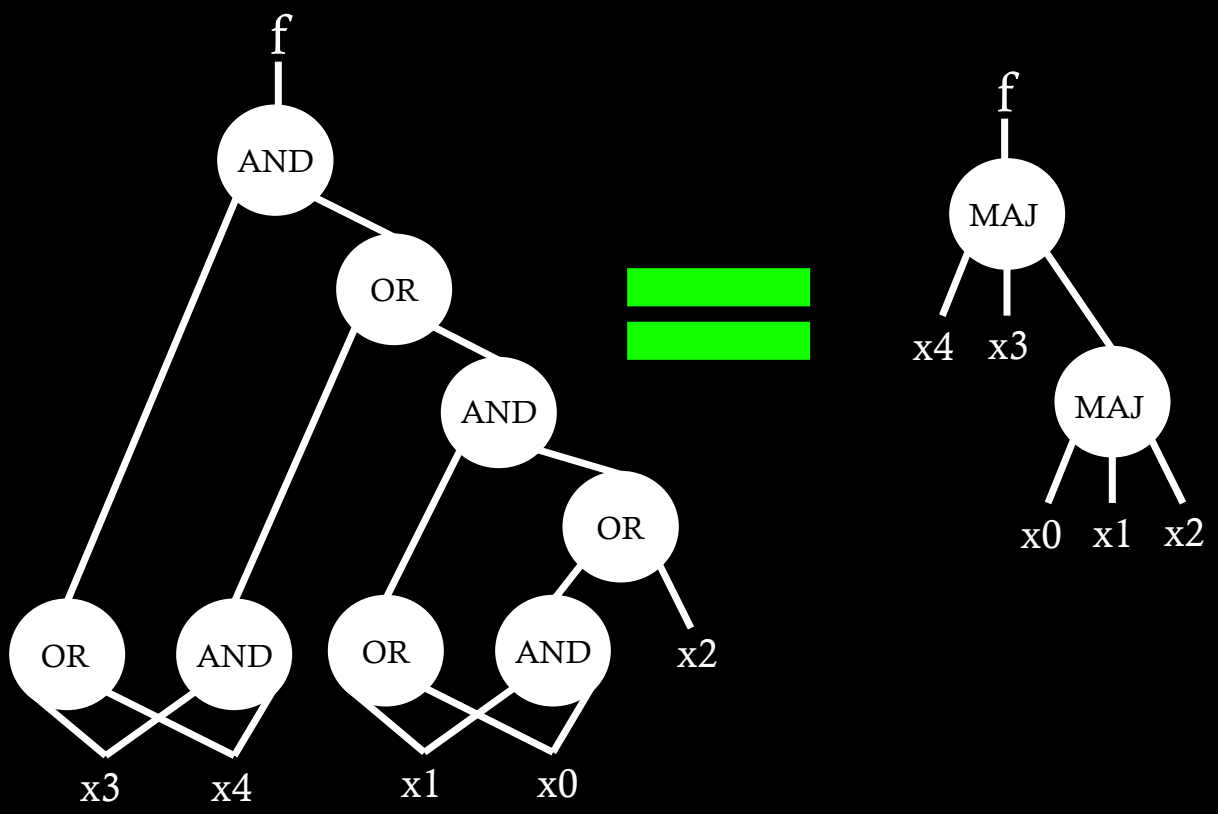
- Majority Logic Synthesis:
  - Why Majority Logic?
  - Majority Inverter Graph (MIG)
  - MIG Optimization
- MIG for Super Conducting Electronics (SCE):
  - SCE Brief Intro
  - Synthesis Challenges for SCE
  - MIG Optimization for SCE
- Conclusions

Outline

- Majority Logic Synthesis:
  - Why Majority Logic?
  - Majority Inverter Graph (MIG)
  - MIG Optimization
- MIG for Super Conducting Electronics (SCE):
  - SCE Brief Intro
  - Synthesis Challenges for SCE
  - MIG Optimization for SCE
- Conclusions

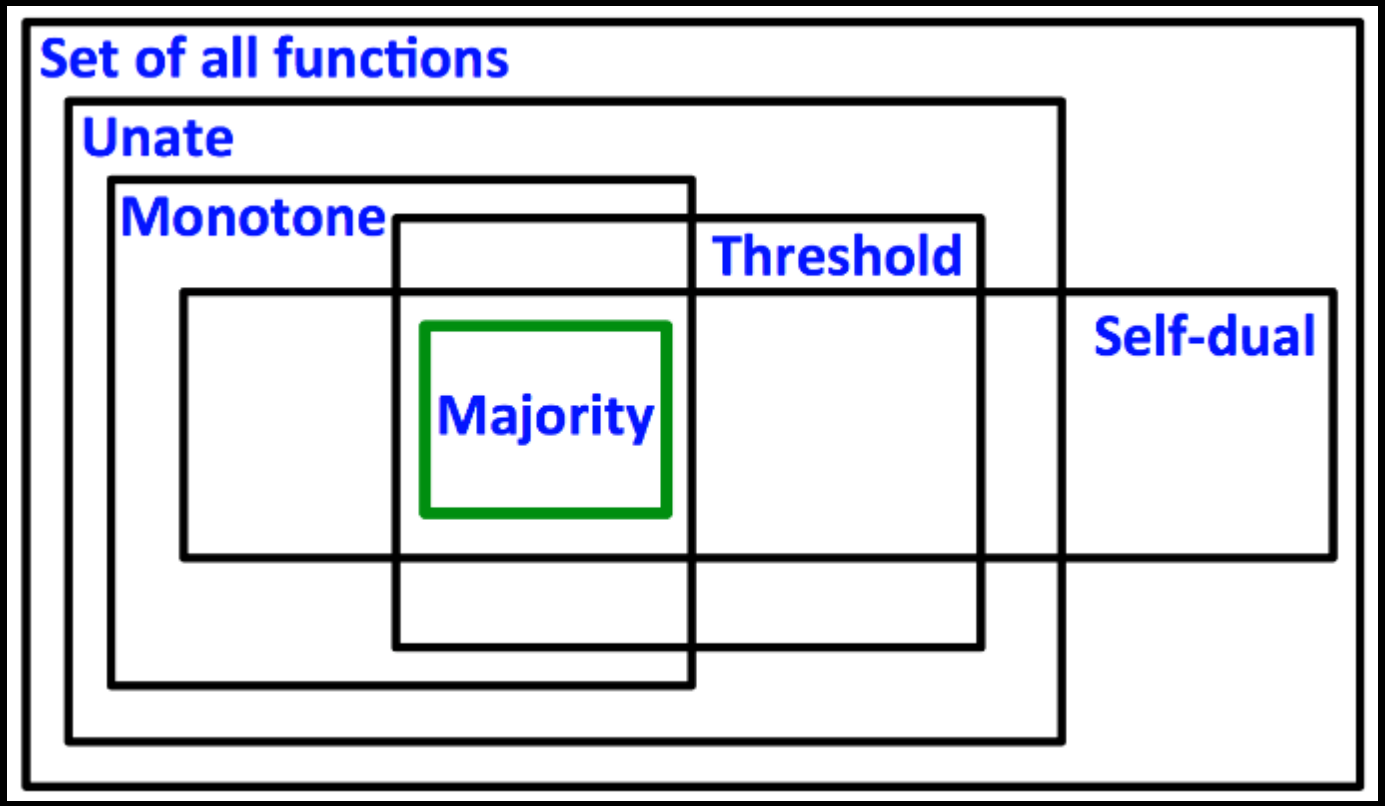
Why Majority Logic?

- Majority logic is a powerful generalization of AND/ORs.
- MAJ(x_1,x_2,x_3,…,x_n)=1 if more than *n/2* inputs are 1.
- MAJ(a,b,c)=ab+ac+bc. MAJ(a,b,1)=a+b. MAJ(a,b,0)=ab.
- More compact as compared to AND-OR logic:



How Powerful is Majority?

- Majority logic vs. AND/OR logic in representing arithmetic circuits.
- Consider small depth representations, target 4/5 logic levels.



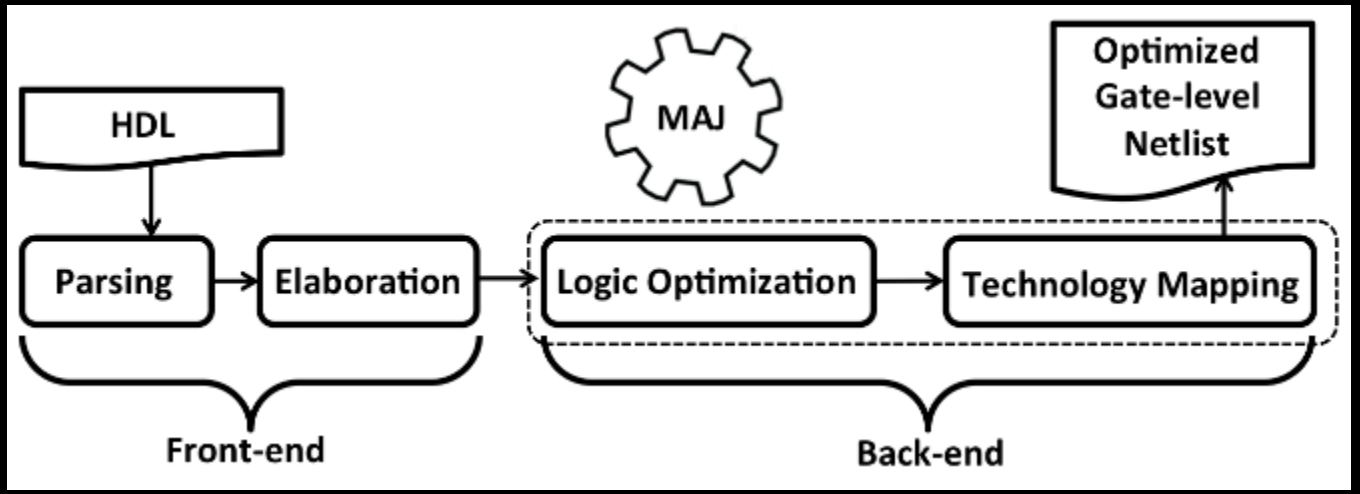
AA. Sherstov, Separating AC 0 from depth-2 majority circuits, Proc. STOC, 2007

Matthias Krause and Pavel Pudlak, On the computational power of depth-2 circuits with threshold and modulo gates, Theor. Comput. Sci., 174 (1997), pp. 137–156.

Kai-Yeung Siu and Vwani P. Roychowdhury, On optimal depth threshold circuits for multiplication and related problems, SIAM J. Discrete Math., 7 (1994), pp. 284–292.

Exploiting Majority Logic

- There is an exponential gap between the expressive power of traditional AND/OR circuits and MAJ circuits when considering arithmetic.
- So, why not exploiting the majority logic representation expressiveness when synthesizing circuits?



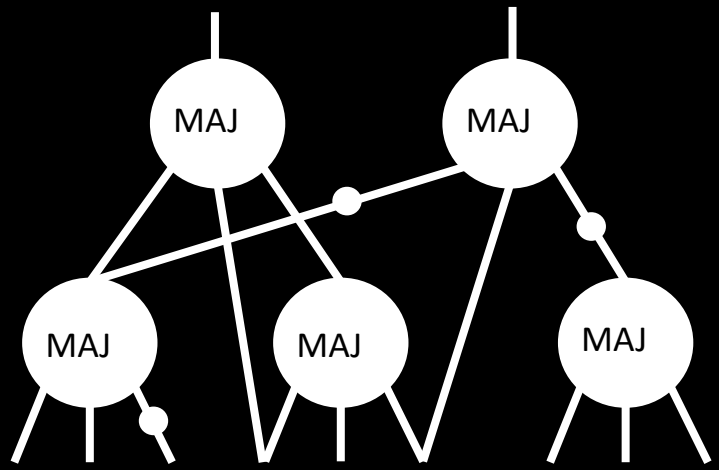
- In order to manipulate majority logic we define a homogenous data structure.
- We call it Majority-Inverter Graph.

Outline

- Majority Logic Synthesis:
  - Why Majority Logic?
  - Majority Inverter Graph (MIG)
  - MIG Optimization
- MIG for Super Conducting Electronics (SCE):
  - SCE Brief Intro
  - Synthesis Challenges for SCE
  - MIG Optimization for SCE
- Conclusions

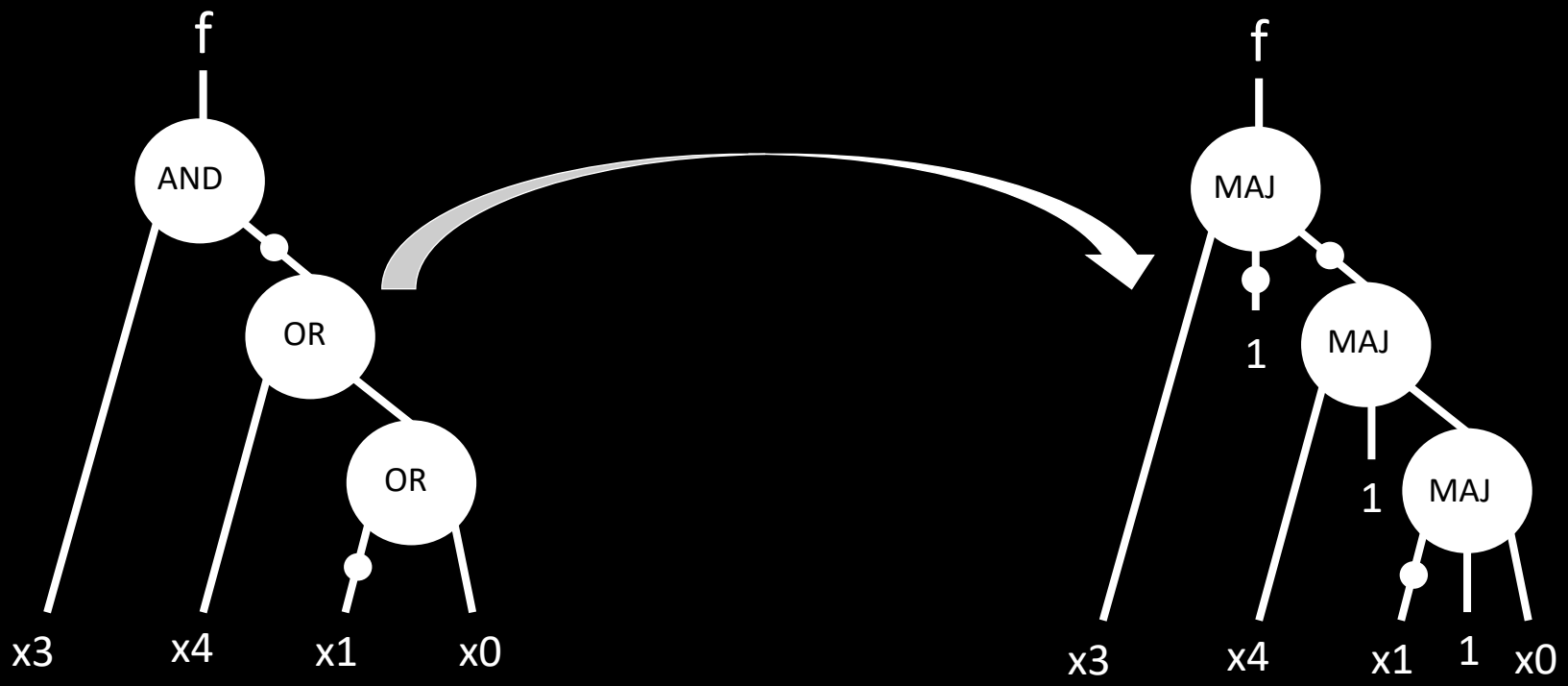
Majority-Inverter Graph

Definition: An MIG is a logic network consisting of 3-input majority nodes and regular/complemented edges.



MIG Properties

AOIGs ➔ MIGs



MIGs **include** AOIGs **include** AIGs

Manipulating MIGs: MIG Boolean Algebra

Ω
1- Commutativity: $M(x, y, z) = M(y, x, z) = M(z, y, x)$
2- Majority: $if(x = y), M(x, y, z) = x = y$
$if(x = y'), M(x, y, z) = z$
3- Associativity: $M(x, u, M(y, u, z)) = M(z, u, M(y, u, x))$
4- Distributivity: $M(x, y, M(u, v, z)) = M(M(x, y, u), M(x, y, v), z)$
5- Inverter Propagation: $M'(x, y, z) = M(x', y', z')$

Theorem: (B,M,',0,1) subject to axiom in Ω is a Boolean algebra

MIG Boolean Algebra

Ω

1- Commutativity: $M(x, y, z) = M(y, x, z) = M(z, y, x)$

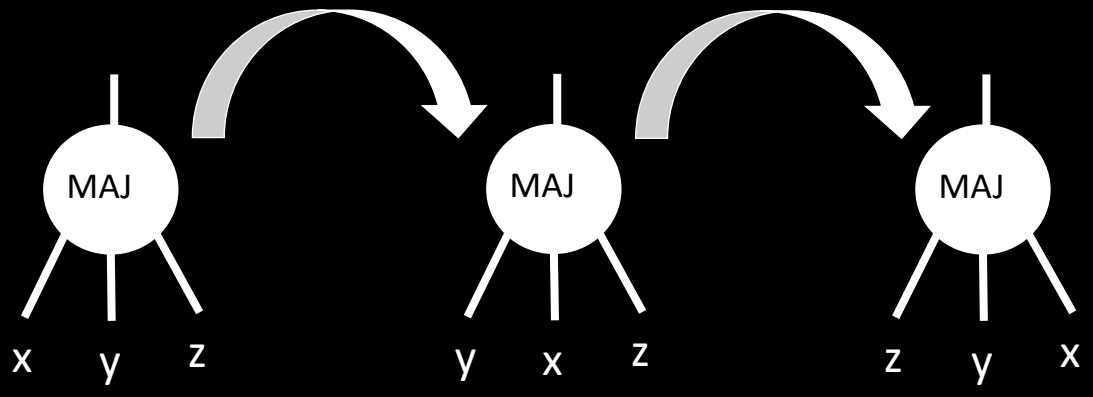
2- Majority: $if(x = y), M(x, y, z) = x = y$

$if(x = y'), M(x, y, z) = z$

3- Associativity: $M(x, u, M(y, u, z)) = M(z, u, M(y, u, x))$

4- Distributivity: $M(x, y, M(u, v, z)) = M(M(x, y, u), M(x, y, v), z)$

5- Inverter Propagation: $M'(x, y, z) = M(x', y', z')$



MIG Boolean Algebra

Ω

1- Commutativity: $M(x, y, z) = M(y, x, z) = M(z, y, x)$

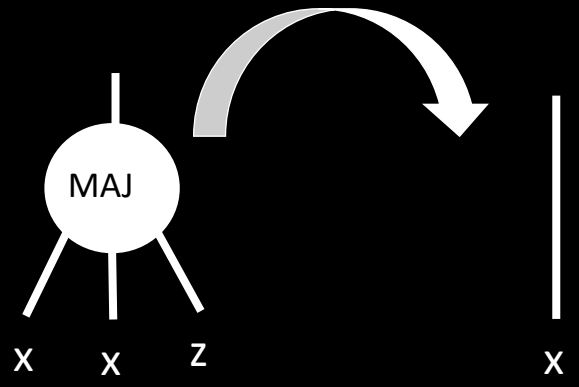
2- Majority: $if(x = y),\ M(x, y, z) = x = y$

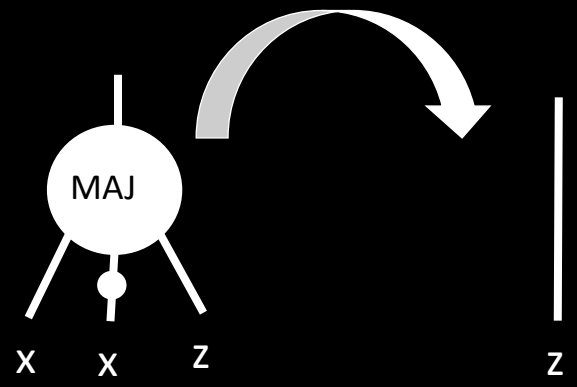
$if(x = y'),\ M(x, y, z) = z$

3- Associativity: $M(x, u, M(y, u, z)) = M(z, u, M(y, u, x))$

4- Distributivity: $M(x, y, M(u, v, z)) = M(M(x, y, u), M(x, y, v), z)$

5- Inverter Propagation: $M'(x, y, z) = M(x', y', z')$





MIG Boolean Algebra

Ω

1- Commutativity: $M(x, y, z) = M(y, x, z) = M(z, y, x)$

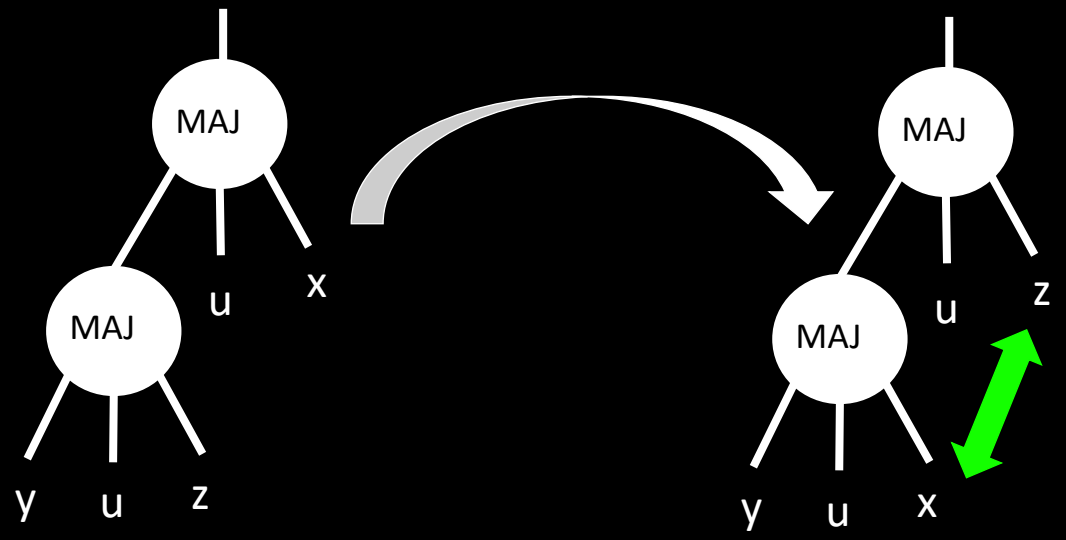
2- Majority: $if(x = y), M(x, y, z) = x = y$

$if(x = y'), M(x, y, z) = z$

3- Associativity: $M(x, u, M(y, u, z)) = M(z, u, M(y, u, x))$

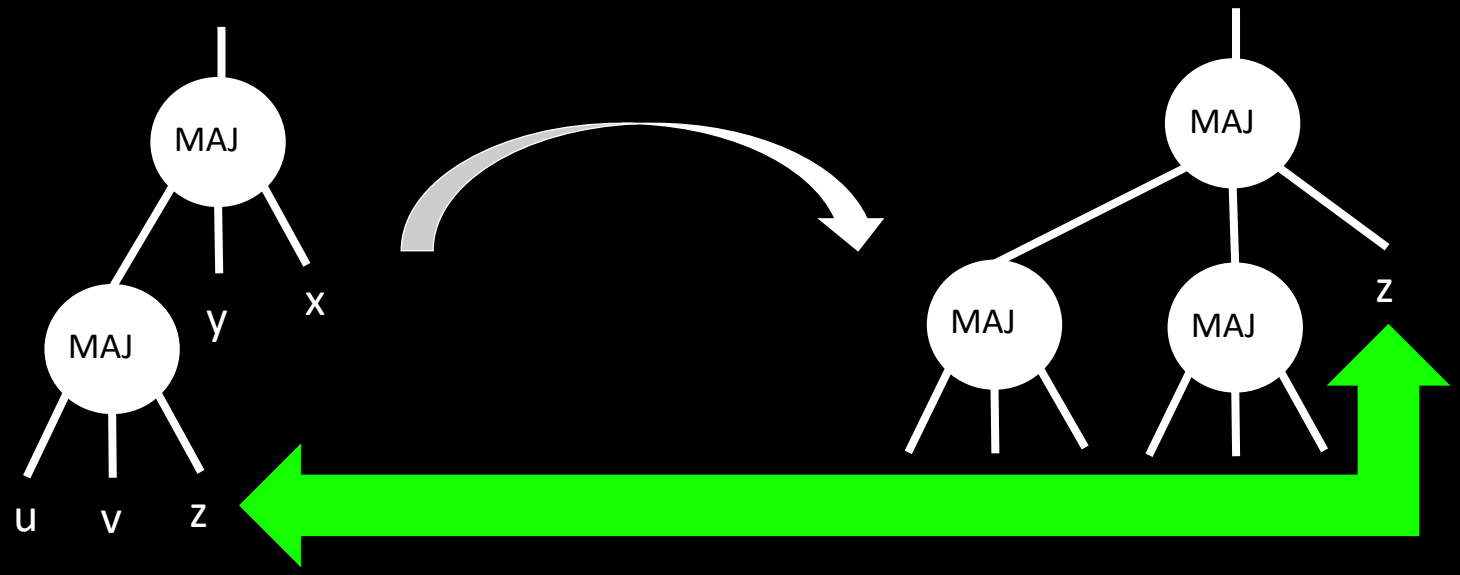
4- Distributivity: $M(x, y, M(u, v, z)) = M(M(x, y, u), M(x, y, v), z)$

5- Inverter Propagation: $M'(x, y, z) = M(x', y', z')$



MIG Boolean Algebra

Ω {
1- Commutativity: $M(x, y, z) = M(y, x, z) = M(z, y, x)$
2- Majority: $if(x = y), M(x, y, z) = x = y$
$if(x = y'), M(x, y, z) = z$
3- Associativity: $M(x, u, M(y, u, z)) = M(z, u, M(y, u, x))$
4- Distributivity: $M(x, y, M(u, v, z)) = M(M(x, y, u), M(x, y, v), z)$
5- Inverter Propagation: $M'(x, y, z) = M(x', y', z')$
}



MIG Boolean Algebra

Ω

1- Commutativity: $M(x, y, z) = M(y, x, z) = M(z, y, x)$

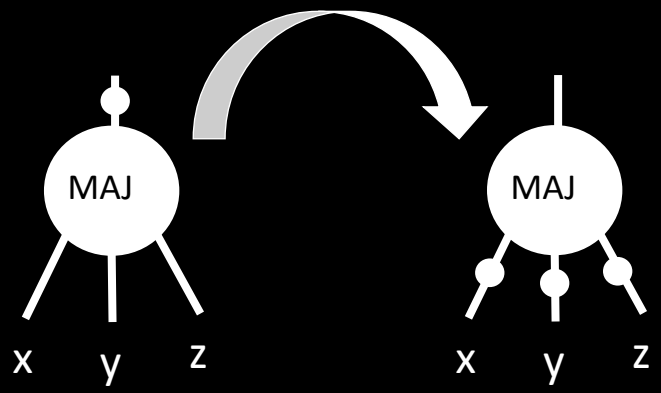
2- Majority: $if(x = y), M(x, y, z) = x = y$

$if(x = y'), M(x, y, z) = z$

3- Associativity: $M(x, u, M(y, u, z)) = M(z, u, M(y, u, x))$

4- Distributivity: $M(x, y, M(u, v, z)) = M(M(x, y, u), M(x, y, v), z)$

5- Inverter Propagation: $M'(x, y, z) = M(x', y', z')$



Outline

- Majority Logic Synthesis:
  - Why Majority Logic?
  - Majority Inverter Graph (MIG)
  - MIG Optimization
- MIG for Super Conducting Electronics (SCE):
  - SCE Brief Intro
  - Synthesis Challenges for SCE
  - MIG Optimization for SCE
- Conclusions

Optimizing MIGs

Ω
1- Commutativity: $M(x, y, z) = M(y, x, z) = M(z, y, x)$
2- Majority: $if(x = y), M(x, y, z) = x = y$
$if(x = y'), M(x, y, z) = z$
3- Associativity: $M(x, u, M(y, u, z)) = M(z, u, M(y, u, x))$
4- Distributivity: $M(x, y, M(u, v, z)) = M(M(x, y, u), M(x, y, v), z)$
5- Inverter Propagation: $M'(x, y, z) = M(x', y', z')$

- Ω is the basis for more elaborated optimization transformations.
- For instance, it is possible to extend associativity:
  - Complementary Associativity:
  - $M(x, u, M(y, u', z)) = M(x, u, M(y, x, z))$

Theorem: MIG Boolean algebra is sound and complete

Optimizing MIGs

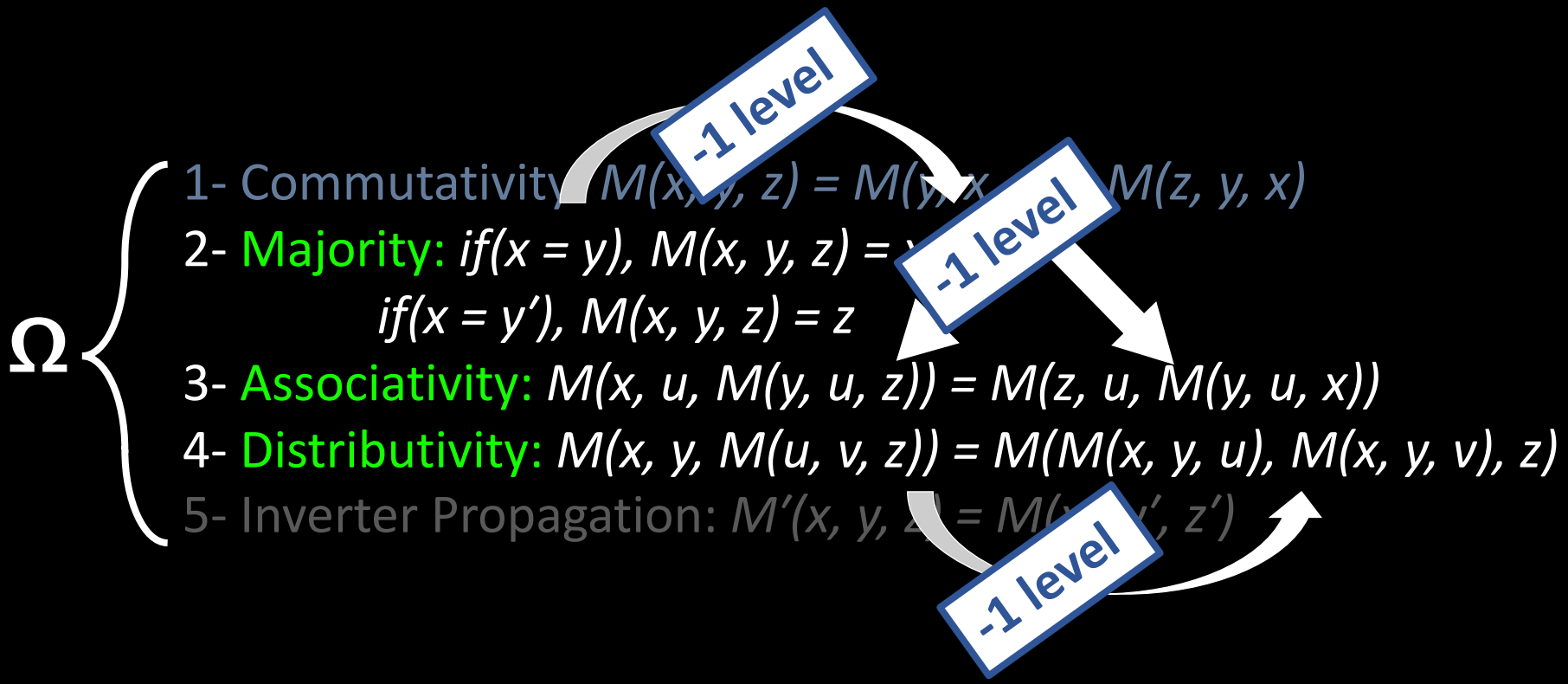
Ω
1- Commutativity: $M(x, y, z) = M(y, x, z) = M(z, y, x)$
2- Majority: $if(x = y), M(x, y, z) = x = y$
$if(x = y'), M(x, y, z) = z$
3- Associativity: $M(x, u, M(y, u, z)) = M(z, u, M(y, u, x))$
4- Distributivity: $M(x, y, M(u, v, z)) = M(M(x, y, u), M(x, y, v), z)$
5- Inverter Propagation: $M'(x, y, z) = M(x', y', z')$

- By using Ω transformations we want to optimize an MIG
- What do we care about?
- **Area**
- **Delay** ➔ MIG size (details in TCAD'16)
- **Power** ➔ MIG depth – discussed in this presentation

➔ MIG SW Activity (details in TCAD'16)

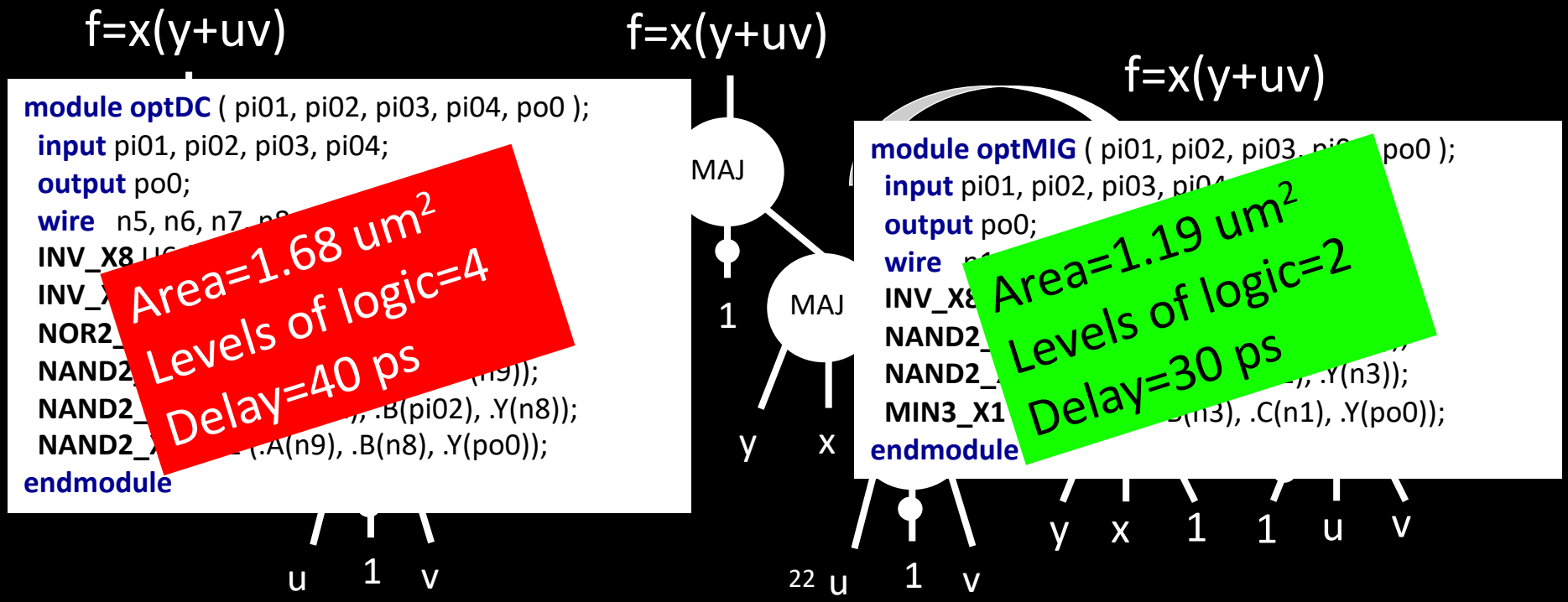
MIG Depth Optimization

- How to reduce the depth of an MIG?
- Let's see what comes handy from Ω:



MIG Depth Optimization

- Rationale: move critical variables closer to the outputs via associativity, distributivity and majority rules
- Reshaping the MIG with other Ω rules

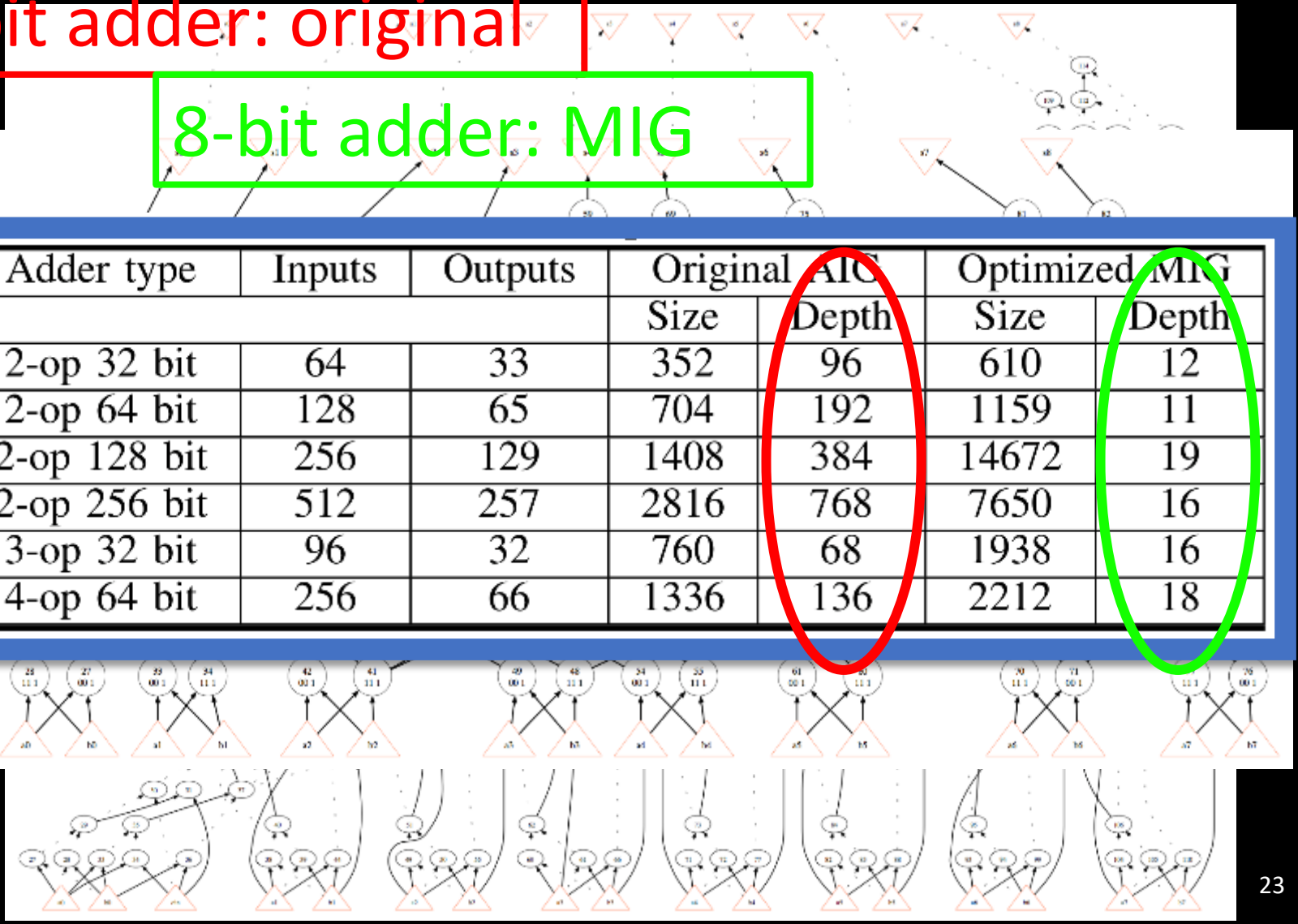


Logic Optimization Experiments: Adders Case Study

8-bit adder: original

8-bit adder: MIG

Adder type	Inputs	Outputs	Original AIG		Optimized MIG	
			Size	Depth	Size	Depth
2-op 32 bit	64	33	352	96	610	12
2-op 64 bit	128	65	704	192	1159	11
2-op 128 bit	256	129	1408	384	14672	19
2-op 256 bit	512	257	2816	768	7650	16
3-op 32 bit	96	32	760	68	1938	16
4-op 64 bit	256	66	1336	136	2212	18



Outline

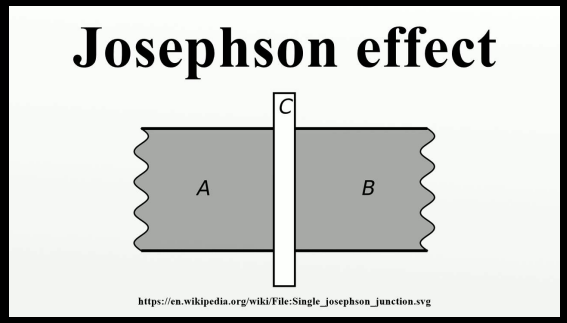
- Majority Logic Synthesis:
  - Why Majority Logic?
  - Majority Inverter Graph (MIG)
  - MIG Optimization
- MIG for Super Conducting Electronics (SCE):
  - SCE Brief Intro
  - Synthesis Challenges for SCE
  - MIG Optimization for SCE
- Conclusions

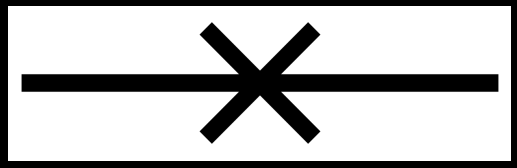
Outline

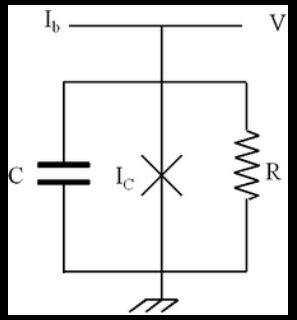
- Majority Logic Synthesis:
  - Why Majority Logic?
  - Majority Inverter Graph (MIG)
  - MIG Optimization
- MIG for Super Conducting Electronics (SCE):
  - SCE Brief Intro
  - Synthesis Challenges for SCE
  - MIG Optimization for SCE
- Conclusions

Super Conducting Electronics

- High level overview of SCE from a synthesis perspective.
- Operation of electronic circuits when superconducting phenomena kick in.
  - Around a few degrees Kelvin.
  - R drops to 0.
  - Quantum effects become fundamental.
- New type of elementary devices:
  - Transistors (CMOS) -> Josephson Junction (SCE).
  - JJ is a 2 terminal device, share some functionality aspect with diodes.
  - Pulse-logic: logic 1 is a pulse, logic 0 is absence of a pulse.

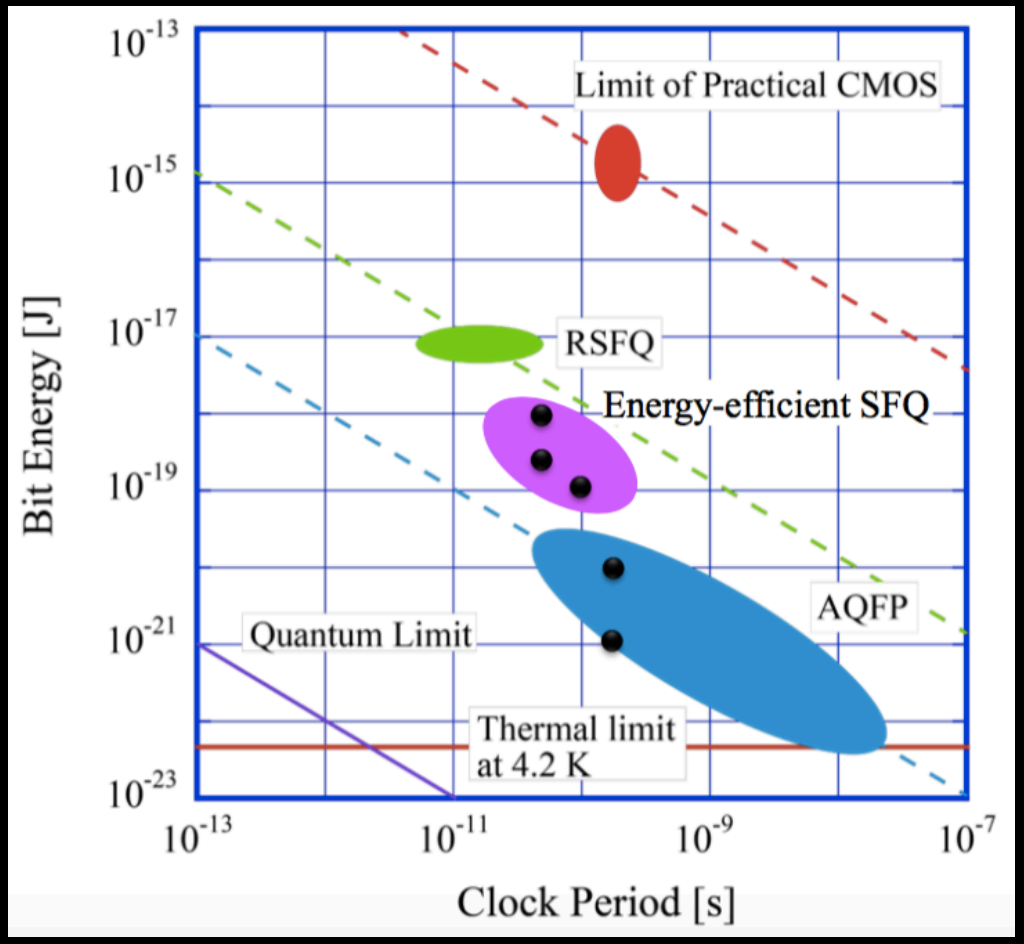






Super Conducting Electronics

- Why SCE?
- Speed
  - Target clock frequencies in the range of tens to hundreds of GHz
- Energy efficiency
  - Close to therm. Limit
- But we need to consider overhead energy to cool down the circuit to a few K.
  - This is not a technology for IoT but for more intensive, high performance, computing applications



N. Yoshikawa et al., "Recent research developments of AQFP toward energy-efficient high-performance computing", EUCAS 2017

Super Conducting Electronics

- US Intelligence Advanced Research Projects Activity (IARPA) **SuperTools** Program:

Synopsys Awarded Multi-Year IARPA SuperTools Contract to Develop EDA Tool Flows for Superconducting Electronics

Program's Goal is to Advance Superconductor Design and Propel Electronics Beyond CMOS

SYNOPSYS®

Silicon to Software™

HYPRES

1850 MELIORA UNIVERSITY of ROCHESTER

Yokohama National University YNU

Stony Brook University

Outline

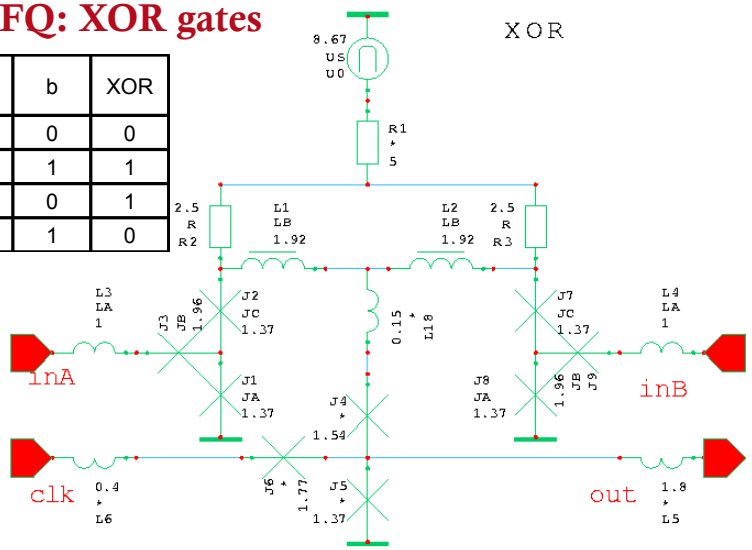
- Majority Logic Synthesis:
  - Why Majority Logic?
  - Majority Inverter Graph (MIG)
  - MIG Optimization
- MIG for Super Conducting Electronics (SCE):
  - SCE Brief Intro
  - Synthesis Challenges for SCE
  - MIG Optimization for SCE
- Conclusions

Challenges in SCE Synthesis: New Devices, New Primitives

- Means for computation: novel set of primitive gates
- Information carried through pulses
- Composition/elimination of pulses give rise to logic interactions

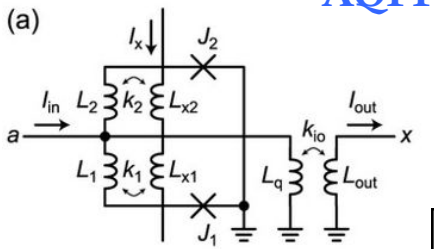
RSFQ: XOR gates

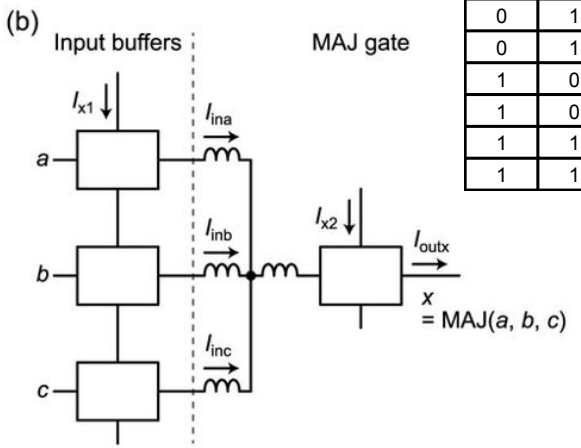
a	b	XOR
0	0	0
0	1	1
1	0	1
1	1	0



Picture courtesy of Stony Brook University: http://www.physics.sunysb.edu/Physics/RSFQ/Lib/AR/xor.html

AQFP: MAJ gates





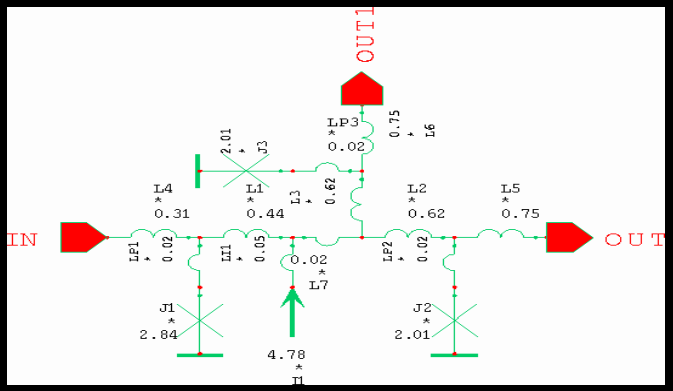
a	b	c	MAJ
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Picture from "Reversible logic gate using adiabatic superconducting devices", Scientific reports, 2014

Challenges in SCE Synthesis: New Constraints and Goals

Fanout restriction

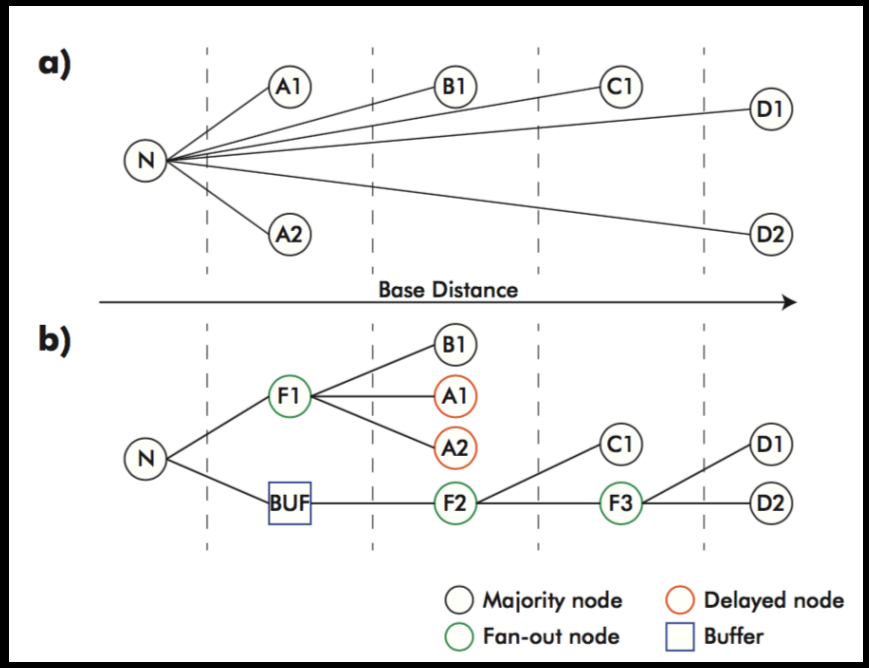
- Combinational gates can have only 1 output
- Special splitter gates to provide multiple fanout



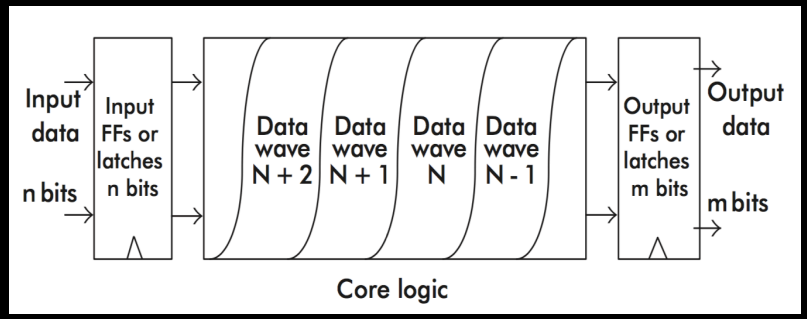
Picture courtesy of Stony Brook University:
http://www.physics.sunysb.edu/Physics/RSFQ/Lib/PB/split.html

Input signals must arrive at the same time

- In order to guarantee correct functionality
- Consequence of the SCE physics & JJ operation
- To address this, all gates are synced with a clock
- Logic signals must arrive in data coherent "waves"
  - Extension of pipelining: wave pipelining



Adapted from "Wave Pipelining for Majority-based Beyond-CMOS Technologies", DATE'17.



Tackle The Challenges: Extend Traditional Synthesis Methods

Gate inputs signals arrive at the same time/clock

- **Wave pipelining – insert clocked buffers**

Fan-out restriction

- **Insertion of splitter trees**

Favor new efficient logic primitives

- **Boolean extraction and native algebras**

Synthesis goals

Minimize # of required buffers

Minimize # of required splitters

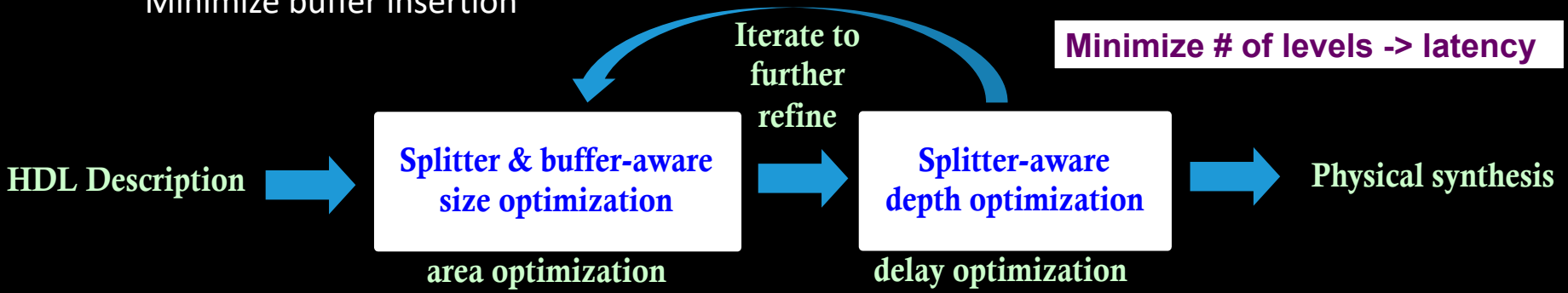
Exploit new gates expressiveness

Empower traditional multi-level synthesis algorithms with this information:

- Area optimization aims at maximizing logic sharing
  - But this creates high fanout gates -> splitter cost
  - Logic optimization techniques, e.g., Kernel extraction, to take into account fanout/splitter cost
- Depth (logic levels) minimization as main timing goal
  - Correlates with latency of computation in gate-clocked scenario
- XOR/MAJ extraction and manipulation
  - XOR methods and MAJ methods in synthesis
- Balancing levels through all paths
  - Minimize buffer insertion

Minimize # of JJs -> area

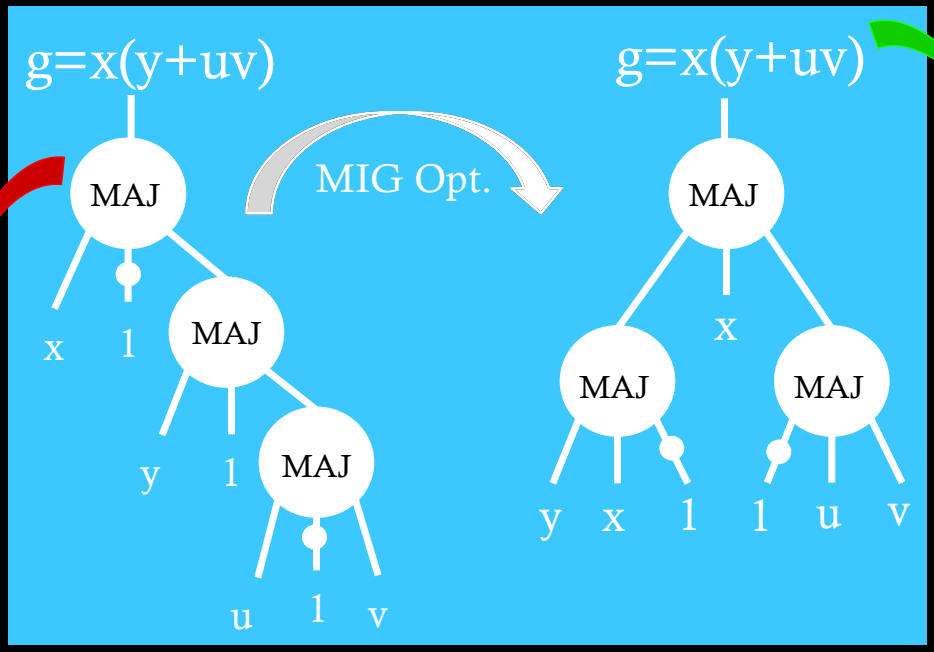
Minimize # of levels -> latency



Outline

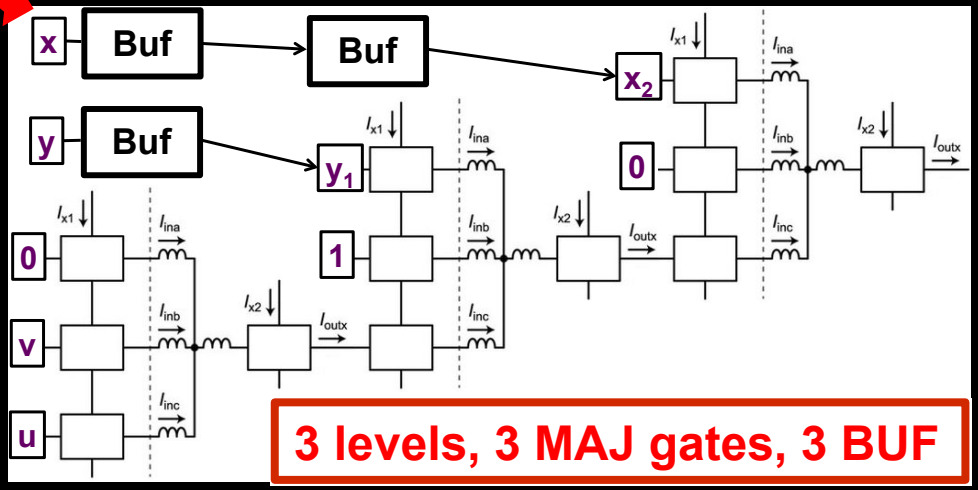
- Majority Logic Synthesis:
  - Why Majority Logic?
  - Majority Inverter Graph (MIG)
  - MIG Optimization
- MIG for Super Conducting Electronics (SCE):
  - SCE Brief Intro
  - Synthesis Challenges for SCE
  - MIG Optimization for SCE
- Conclusions

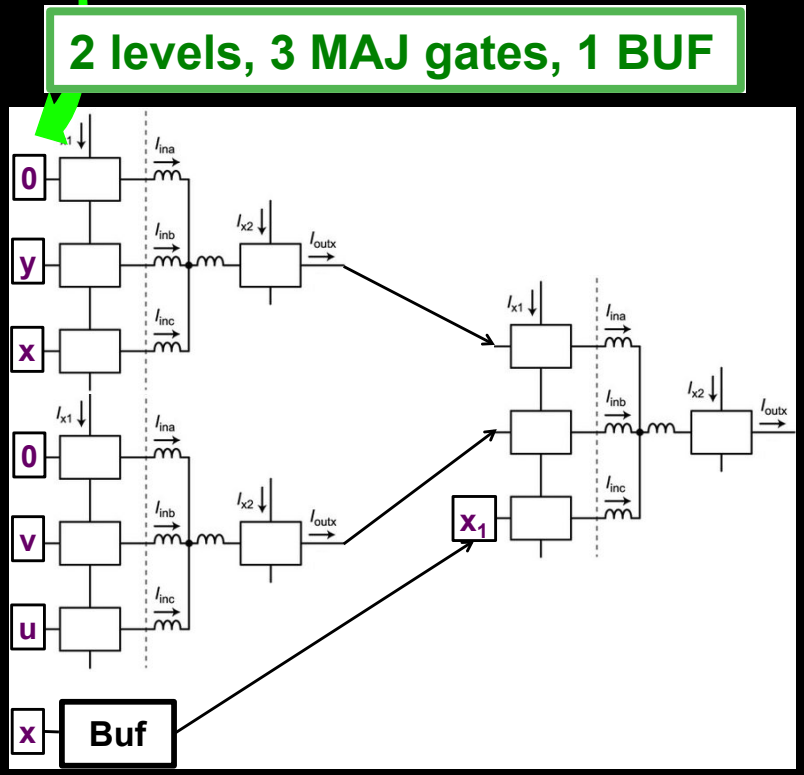
Synthesis Opportunity for AQFP: Majority-Inverter Graphs



1-1 correspondence with AQFP logic primitives

Native Boolean algebra to manipulate MIG, thus optimizing AQFP circuits





Initial Synthesis Evaluations

Initial focus on a small module of a complete processor

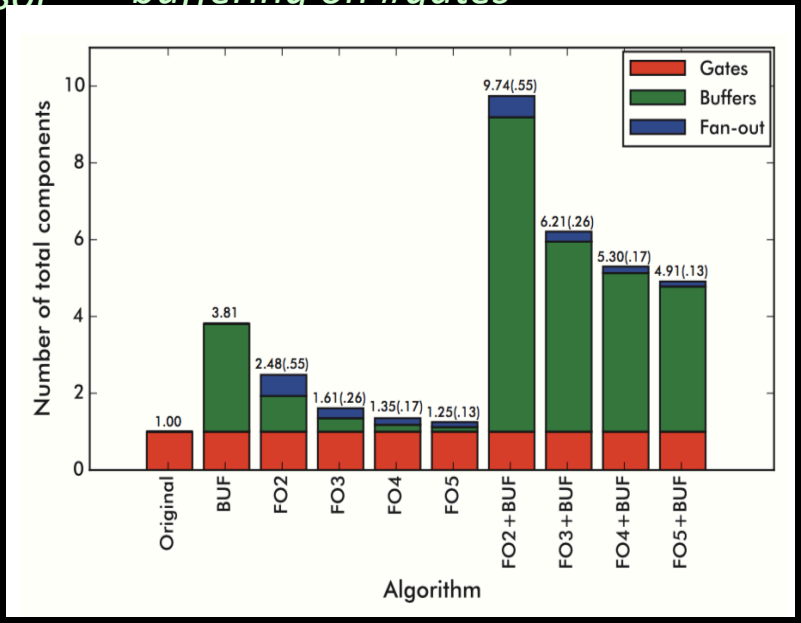
Decoder block, initial logic chars:
1.5k equivalent AND-2 gates.
27 levels of logic (excluding inverters)
Max fanout for individual gate ~90

We adapted our opt. engines to (i) reduce max fanout and (ii) reduce levels of logic. These two targets don't go together, making optimization difficult: we look for a tradeoff
Best (minimum) max fanout: 28
Best (minimum) # of levels: 19

Our chosen tradeoff, after splitter and buffer insertion, with RSFQ technology considerations, produced a circuit with:
3k equivalent gates.
31 levels of logic.

3-output splitter insertion and buffering on MIG, for seven academic benchmarks

Evaluate the separate/composite the impact of splitter insertion and buffering on #gates



	Depth		Size	
	Original	WP	Original	WP
SASC	6	9	622	1885
DES AREA	22	38	4187	13325
MUL32	36	58	9097	18998
HAMMING	61	96	2072	11523
MUL64	109	135	25773	139914
REVX	143	225	7517	34911
DIFFEQ1	219	282	17726	306937

Evaluation data adapted from "Wave Pipelining for Majority-based Beyond-CMOS Technologies", DATE'17.

Outline

- Majority Logic Synthesis:
  - Why Majority Logic?
  - Majority Inverter Graph (MIG)
  - MIG Optimization
- MIG for Super Conducting Electronics (SCE):
  - SCE Brief Intro
  - Synthesis Challenges for SCE
  - MIG Optimization for SCE
- Conclusions

Conclusions

- Majority-Inverter Graphs support optimization techniques.
  - The expressive power of MIG Boolean algebra axioms, such as distributivity and inverter propagation, permits more agile logic manipulation.

- MIG optimization show promising results.
  - MIG can improve QoR for CMOS design flows.
    - ASICs.
    - FPGAs.
  - MIG are key to enable majority-based emerging nanotechnologies.
    - QCA, SWD, SiNWs, Graphene, etc.
    - MIG are key to design efficiently logic families in SCE, such as AQFP, RQL, etc.

Questions?

Thank you for your attention!