

Logic Synthesis for Secure Computation

Giovanni De Micheli

EPFL

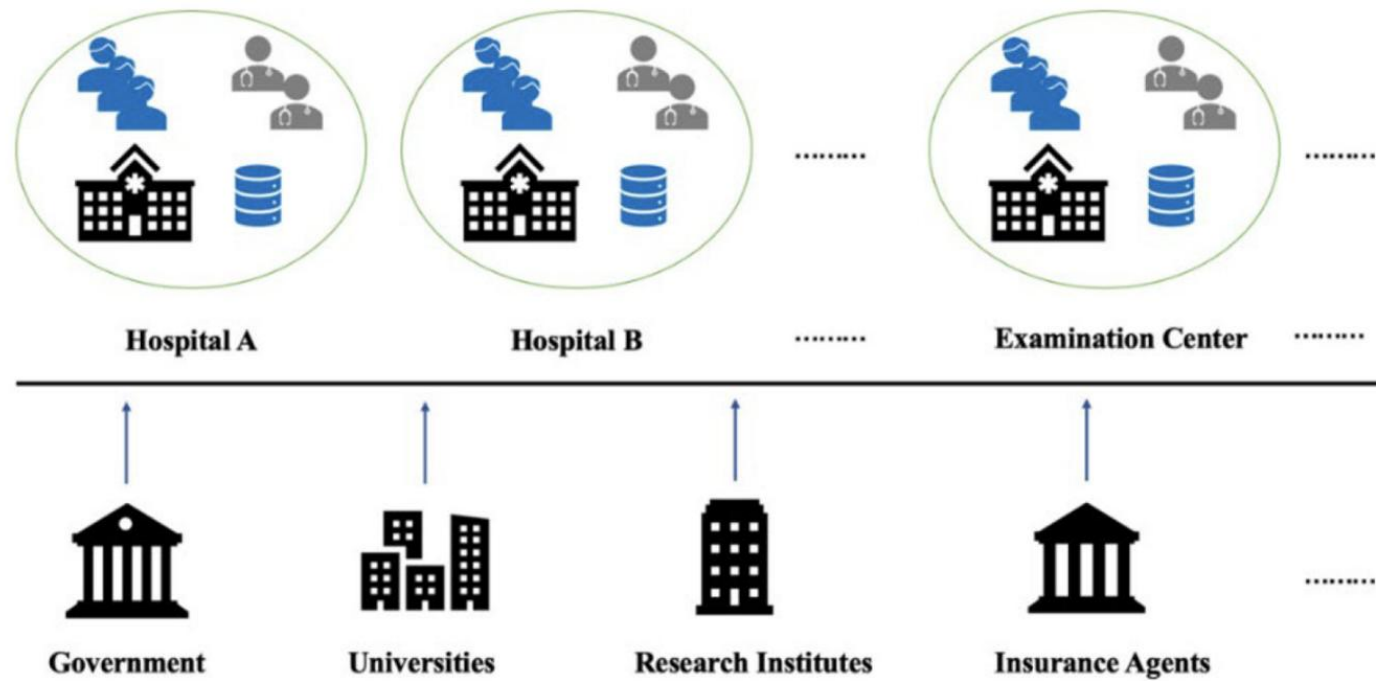


Data is the oil of the 21st century

- Very high value associated with aggregate and personal data
 - Main commodity to steer economy, trade and politics
 - Strong incentive to acquire data by any means
- Data exchange is required in most daily activities
 - Privacy protection is required but has a cost
- Some current data protection schemes are vulnerable
 - Quantum computing technology can break some crypto schemes

Privacy-Preserving Computation and Communication

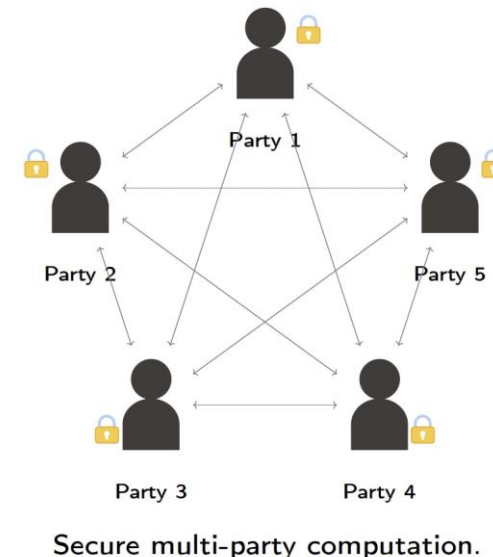
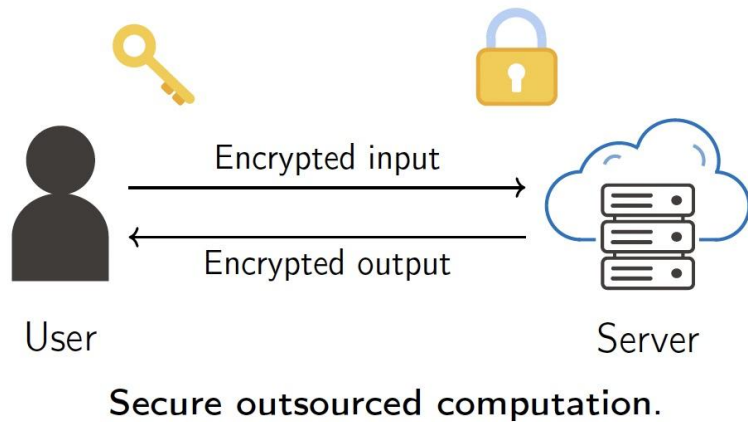
Encourages data-sharing-based collaborations: *federated learning*



A Typical Medical Data Sharing Practice

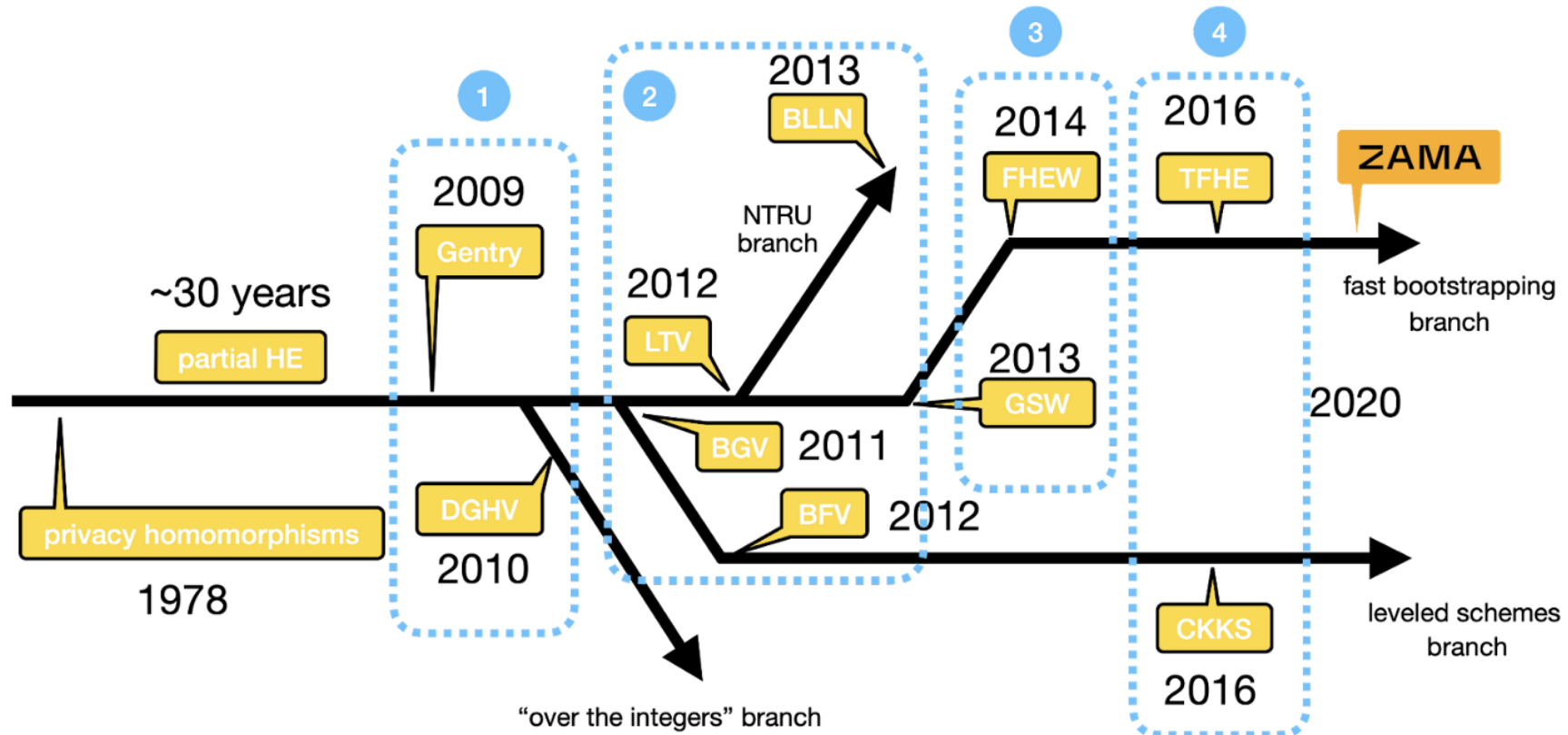
Secure Computation

- Secure outsourced computation (SOC):
 - A client encrypts data and lets an untrusted cloud perform the computation
- Secure multi-party computation (SMPC):
 - Multiple parties jointly compute a function without revealing their private inputs



Fully Homomorphic Encryption (FHE): A Game-Changer

- Perform any computation directly on ciphertexts without ever decrypting the data
- Input (plain.) → Encrypt → Compute on ciphertexts → Decrypt → Output (plain.)



Modern FHE Schemes: Levelled vs. Fast-Bootstrapping

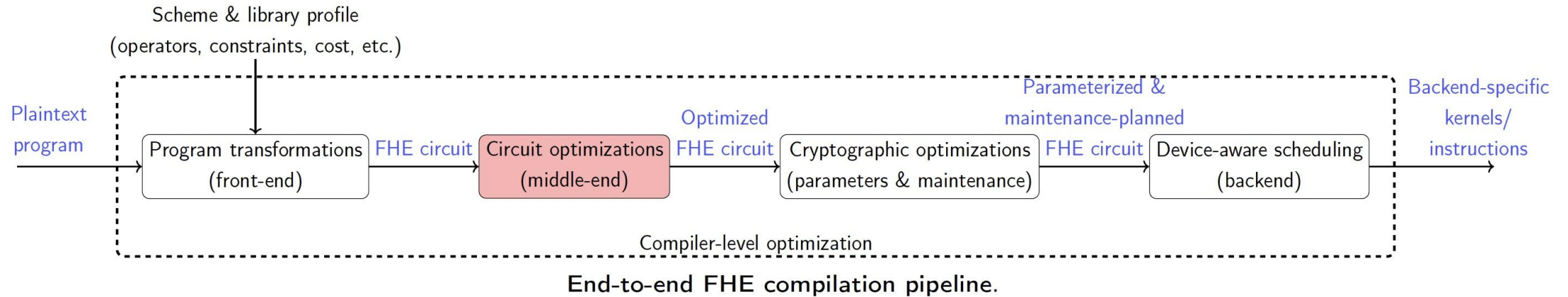
- Modern FHE schemes hide data by embedding small random noise
 - As computations proceed, this noise grows
 - If noise becomes too large, the ciphertext can no longer be correctly decrypted
- Bootstrapping is the operation that “refreshes” a ciphertext
- Depending on how noise is managed, FHE schemes fall into two groups:
 - Levelled schemes — limited depth without bootstrapping (slow)
 - Fast-Bootstrapping schemes — Programmable bootstrapping (PBS):
(1) efficient and (2) simultaneously evaluates a small function
- For both families, logic synthesis can help!

Modern FHE Schemes: Levelled vs. Fast-Bootstrapping

- Levelled schemes (BGV [1], BFV [2], CKKS [3], etc.):
 - Features: Levelled; Slow bootstrapping ($\sim 10\text{--}30\text{s}$); Word-level parallelism.
 - Applications: Arithmetic functions over large integers or even floating-point; Data-parallel workloads.
- *Fast bootstrapping schemes* (FHEW [4], FHE over the torus (TFHE) [5])
 - Features: Efficient bootstrapping ($\sim 5\text{--}50\text{ms}$) that involves a function evaluation.
 - Application: Boolean functions; Arithmetic functions over small integers.

1. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. “(Leveled) Fully Homomorphic Encryption without Bootstrapping”. In: Innovations in Theoretical Computer Science Conference. 2012
2. J. Fan and F. Vercauteren. Somewhat Practical Fully Homomorphic Encryption. Cryptology ePrint Archive, Paper 2012/144. 2012
3. J. H. Cheon et al. “Homomorphic Encryption for Arithmetic of Approximate Numbers”. In: ASIACRYPT. 2017
4. L. Ducas and D. Micciancio. “FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second”. In: EUROCRYPT. 2015
5. I. Chillotti et al. “TFHE: Fast Fully Homomorphic Encryption Over the Torus”. In: Journal of Cryptology (2020)

The Full Stack and The Compiler Level

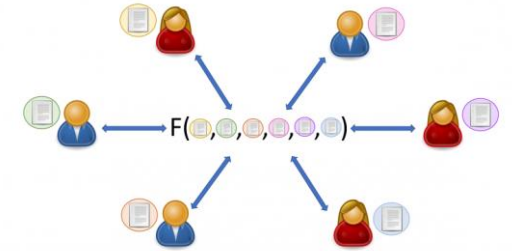


- Improving the efficiency of FHE-based secure computation involves coordinated efforts across multiple layers of the system stack
- Compiler-level goal:
 - Optimize FHE circuit structure, to speed up secure Boolean computation

FHE Status in 2026

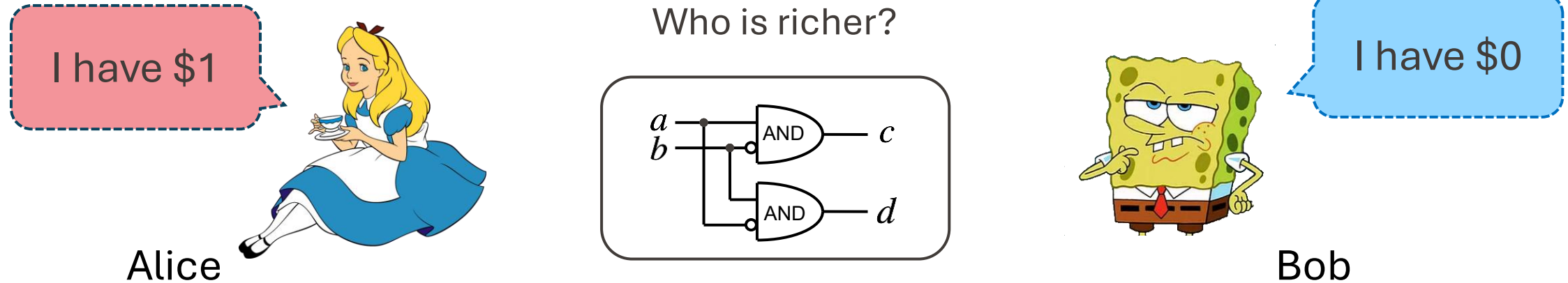
- Large academic and industrial attention
 - New ventures: Corami, Belfort, Zama, ...
- Most efforts attack one piece of the puzzle
 - Need for comprehensive system-level solutions
- Performance and energy costs are still a road block
 - Digital design tools and algorithms are promising

Multi-party secure computation

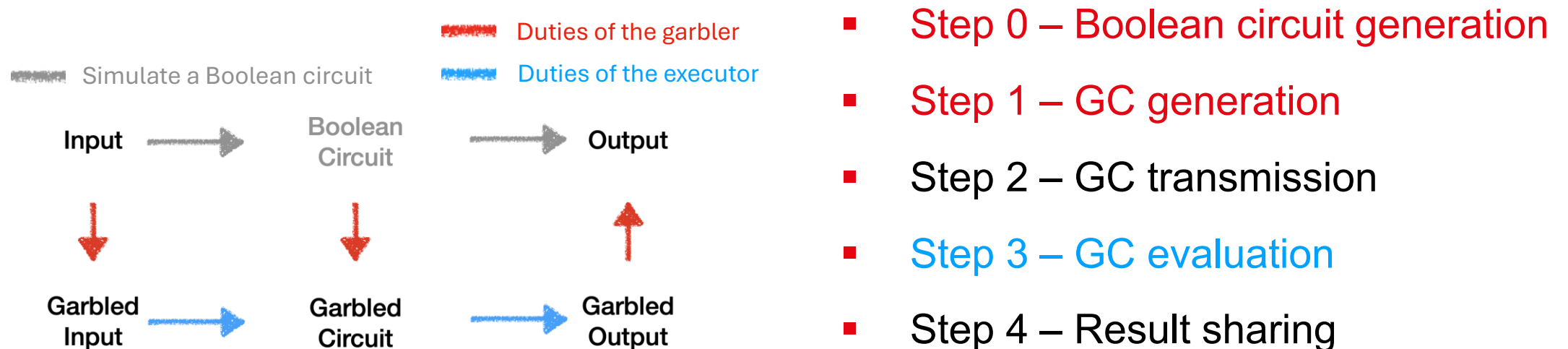


- Compute result from two or more sources
 - Without revealing the source content to each other
- Andrew Yao's garbled circuit protocol [1986]
 - Evaluate a *function* over private inputs without a *trusted third party*
 - The function is described as a *logic network*
- Garbled circuits are secure against a *semi-honest adversary*
 - Curious but following the protocol

Yao's Millionaires' Problem

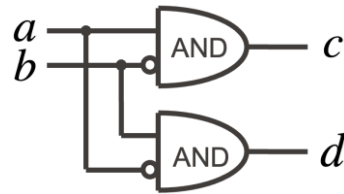


- An overview of the GC protocol:



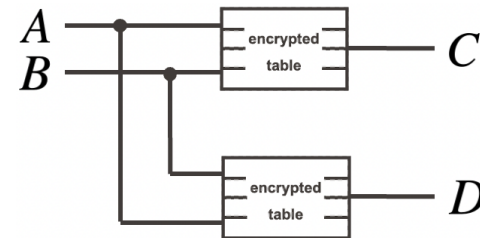
The Garbled Circuits (GC) Protocol

- Analogy between a Boolean circuit and a GC
 - Assume the target computation is a one-bit comparison...



Boolean Circuit

Executing a Boolean circuit:
Boolean operations



Garbled Circuit

Executing a garbled circuit:
look-ups & decrypts

- Excessive **data communication** bottlenecks system performance...
 - Resulted by the need of sending encrypted tables
 - 128 MB data for a 3-layer neural network^[1]

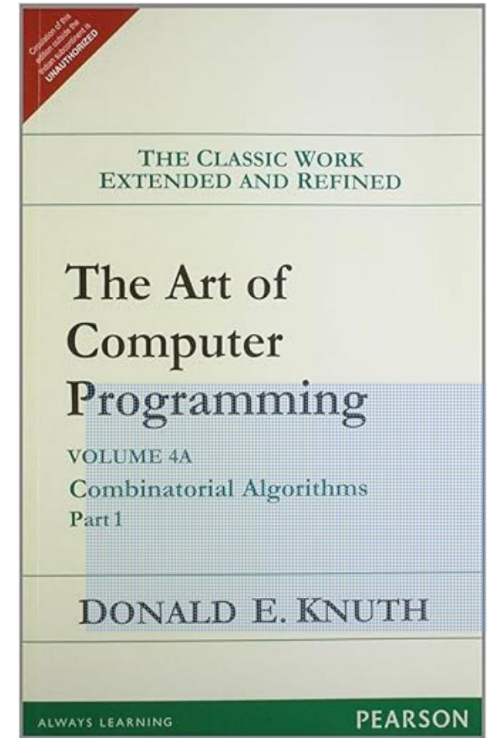
[1] M. Ball, *et al.*, “Garbled neural networks are practical,” in *Cryptology ePrint Archive*, 2019.

GC Status in 2026

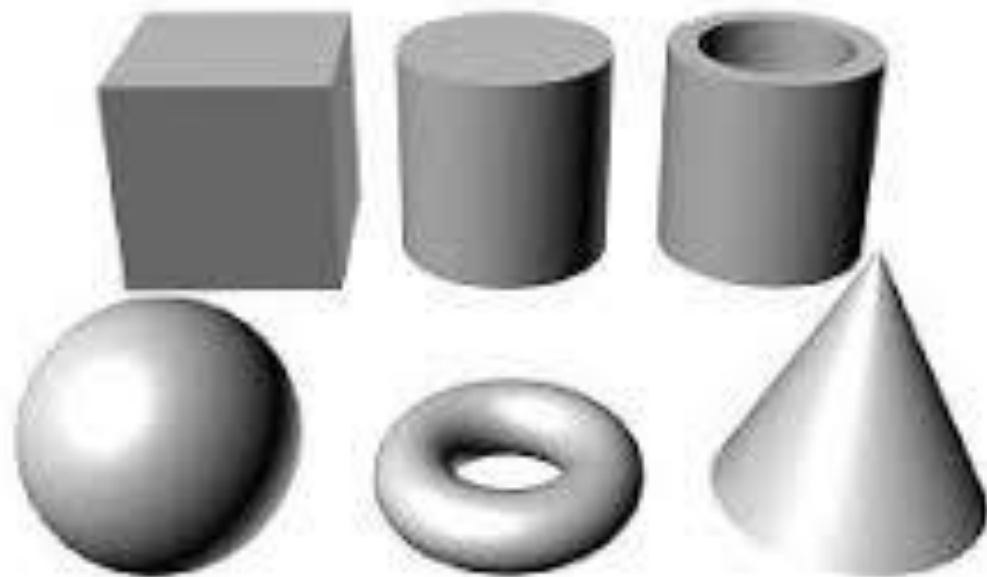
- Renewed interest in GC protocol
 - Financial data analysis, Healthcare monitoring, Blockchain
- Open issues relate to:
 - Resilience to side channel attacks
 - Reducing garbling to the essential shared set
- Small research community but promising large impact

What is a circuit ?

- Interconnection of electrical elements
- Interconnection of logic primitives
- Abstraction of a computation
- Data structures for logic networks
 - Networks of NAND2 and/or NOR2
 - AIG: AND-INVERTER network: ABC [Mishchenko]
 - MIG: Majority-Inverter graph
 - Many other models



What about other primitives?



3-input gate primitives

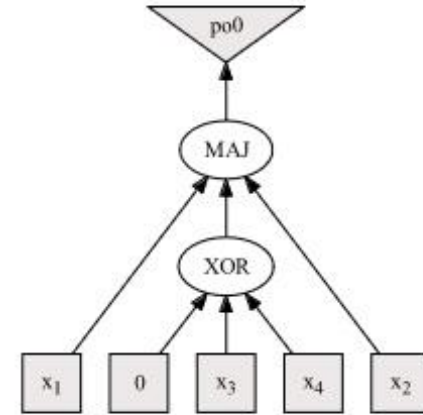
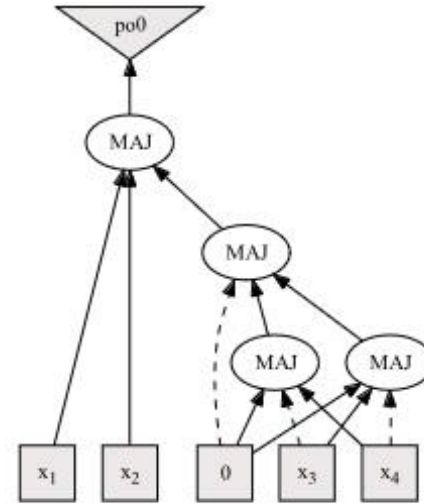
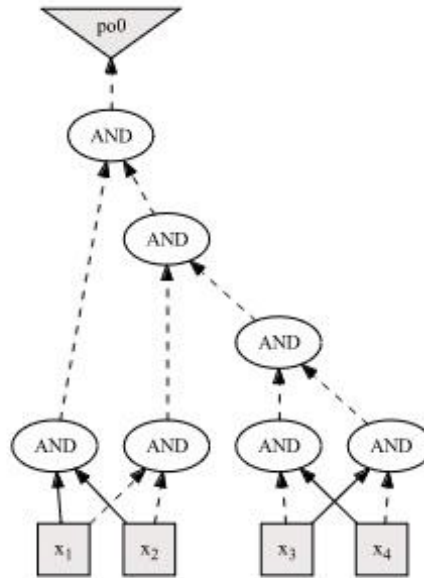
- Explore all 3-input gates + INVerter

Class	Expression	Function	Cube	Name
#1a	$x \oplus (y \oplus x \cdot y)$	#d2		Dot
#1b	$x ? y : z$	#d8		Mux
#1e	$x \oplus yz$	#6a		AndXor
#09	$x \oplus y \oplus z$	#96		Xor3

Exact synthesis for all functions up to 5 inputs and frequent 6 input functions show that **dot** outperforms other primitives

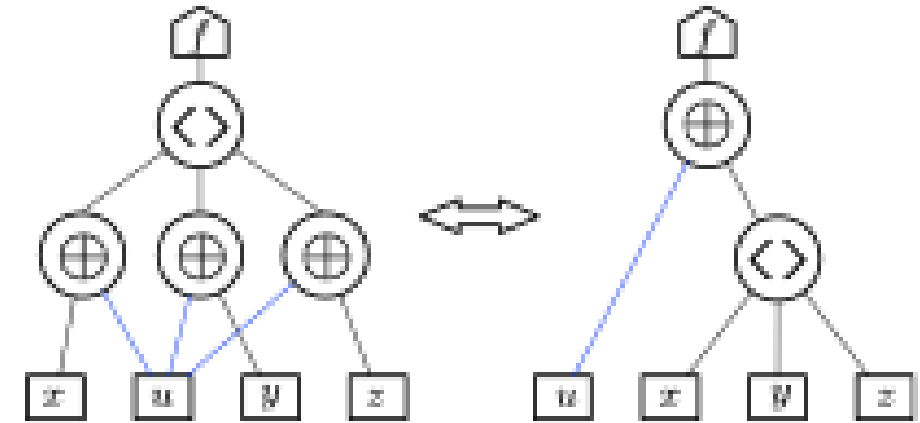
The role of *linearity*

- AND-OR logic is non-linear
- MAJority logic is non-linear
- EXOR are linear operators
 - EXOR play a leading role in realizing adders and multipliers
 - Adders require at least one AND gate per bit
- It is useful to decompose functions into linear and non-linear parts

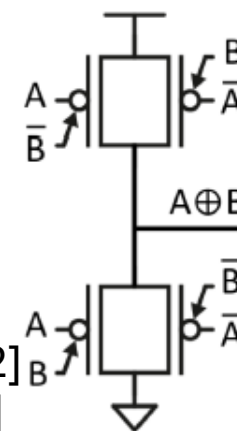


EXOR-MAJORITY models

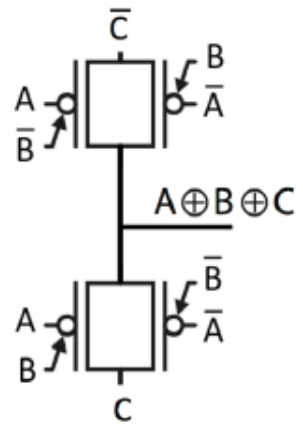
- Use linear and non-linear gates
- XOR – MAJority graph
 - Extension of MIGs (replace INVerter by XOR)
- Self-duality of XORs (odd) and Majority
 - XOR can “flow through” MAJ nodes
- Data structure for controlled-polarity switches
 - Silicon *controlled polarity* gates
 - Nano-relays
 - Optical gates
- **Optimization very successful also for CMOS**



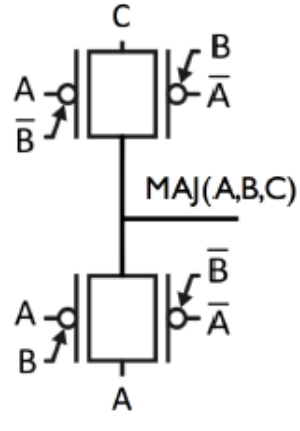
2-input XOR



3-input XOR



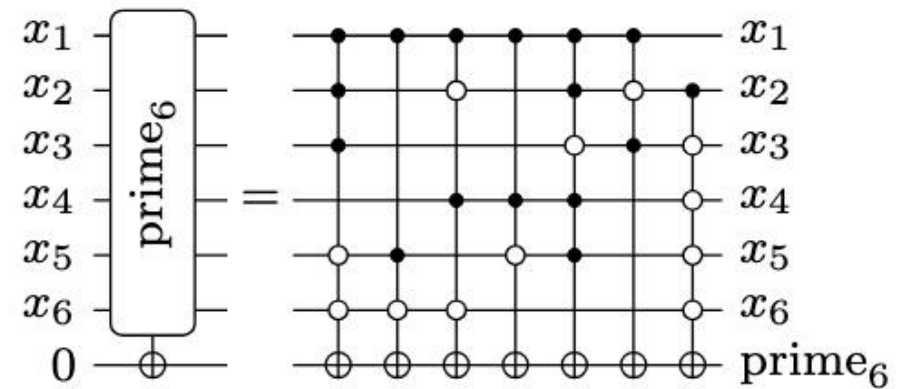
Majority 3



[De Marchi. IEDM 12]
[Rai et al., TCAD 22]

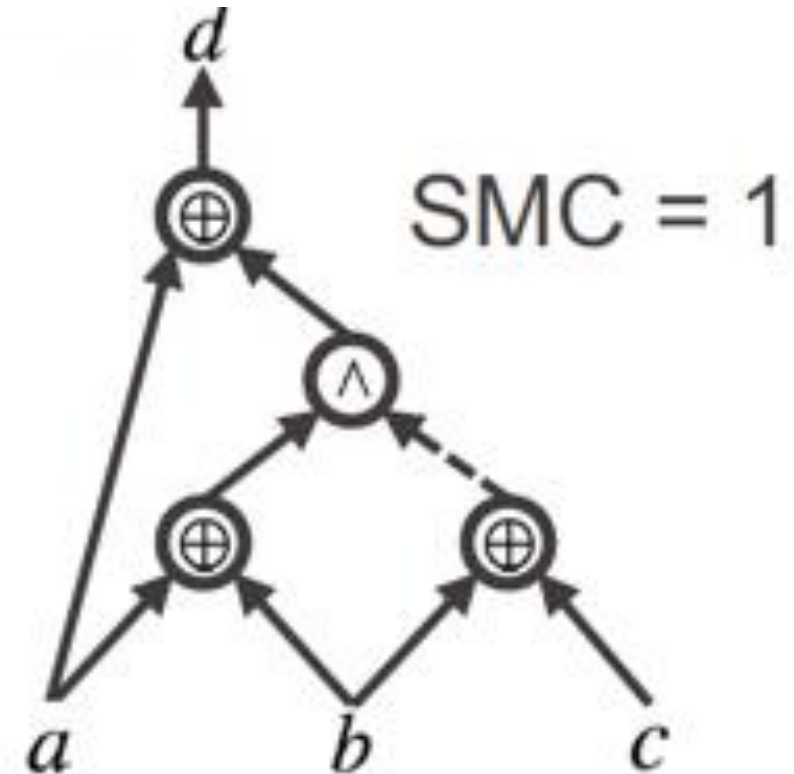
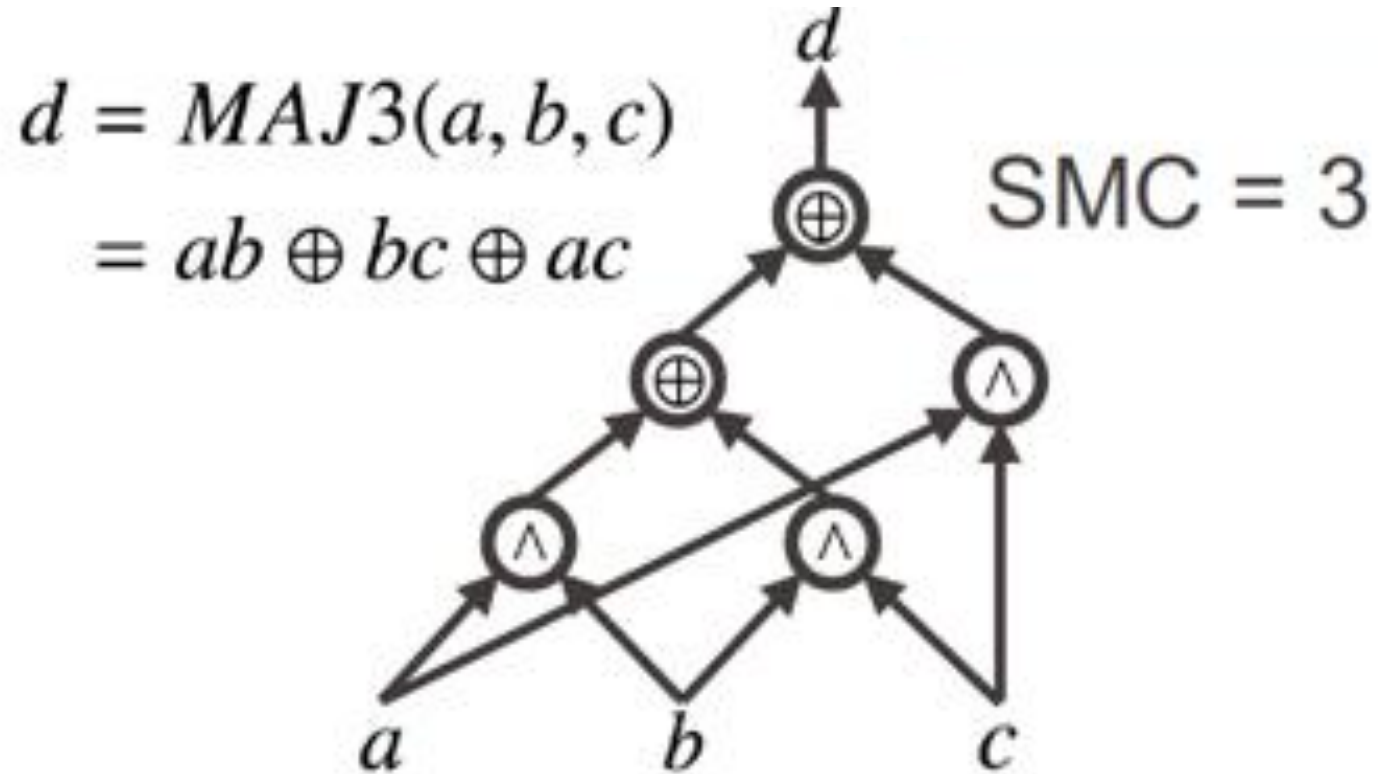
Exclusive sum of products forms

- ESOPs widely studied in logic synthesis
 - Limited application in CMOS due to gate complexity
- ESOP optimization very useful in Quantum Compilation
 - Decomposition into Toffoli gates



[Meuli et al 2018]

The role of *multiplicative complexity*



Applications of XAGs - FHE

- Fully Homomorphic Encryption (FHE)
 - Compute functions using encrypted data
 - Speed of computation is essential
- Apply logic synthesis to restructure computation
 - While keeping noise under a threshold level
- Key metrics:
 - Multiplicative complexity – multipliers generate noise
 - Multiplicative depth – critical path
- Approaches:
 - MC-aware MD optimization ($MC * MD^2$)
 - ESOP balancing



[Yu, WAHC 24]

Applications of XAGs - GC

- Model garbled circuits by XAGs
 - Cost of XOR is null
 - The nonlinear (AND) component is needed
 - Overall GC cost relates to multiplicative complexity
- Additional efficiency can be achieved by changing model
 - Use OneHot gates instead of AND gates
 - OneHot gates express nonlinearity with lower garbling cost
- XAGs can be easily converted to X1Gs: OneHot–Inv graphs
 - More compact and faster GC can be obtained

[Yu, Cryptography 23]

Conclusions

- The unprecedented growth of computing is challenged:
 - Excessive energy consumption
 - Privacy of data
- Logic synthesis and optimization methods can help
 - *Linearity extraction* and *multiplicative complexity* reduction are key to solving various problems
 - Critical path minimization
 - Complexity minimization
- The current large investments in AI/ML software need to be matched by equivalent strong efforts in hardware design
 - Low energy and secure design methods and tools are required

Thank You



[Egylstadir, Iceland. Modern abstract art in nature]