Garbled Circuits in the Cloud using FPGA Enabled Nodes

Kai Huang, Mehmet Gungor, Stratis Ioannidis, Miriam Leeser

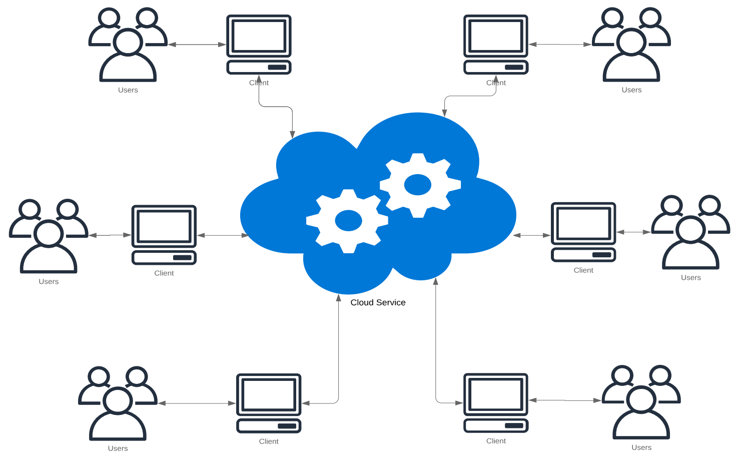
Dept. of Electrical and Computer Engineering
Northeastern University
Boston, MA

Xin Fang

Qualcomm
Boxborough, MA

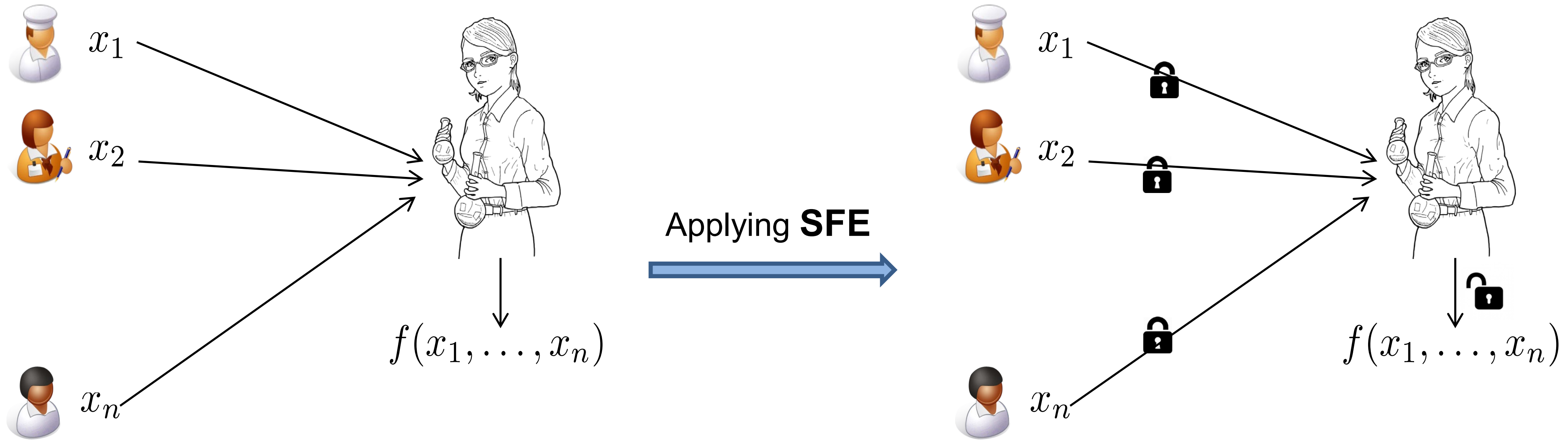
Introduction and Motivation

- More and more computations are done in the cloud with user data
- Secure Function Evaluation (SFE) is needed to protect privacy of user data
- Cloud services provide FPGA infrastructure
- We accelerate garbled circuits in the cloud using FPGA



Northeastern

Secure Function Evaluation

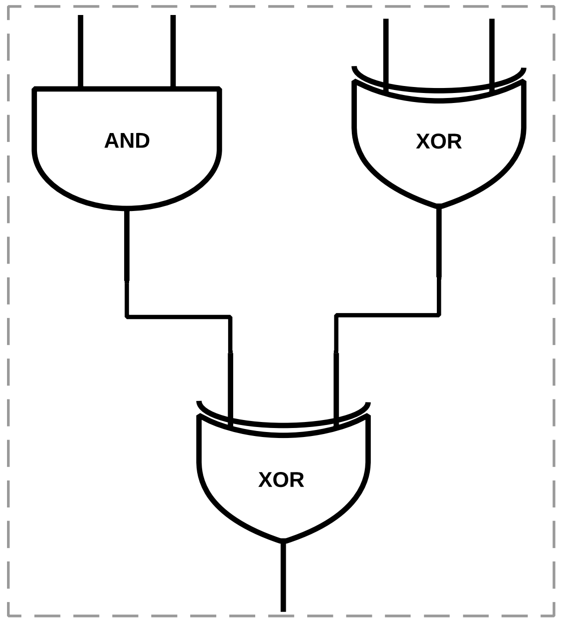


- Only Users have access to their own unencrypted data
- Analyst processes the encrypted data

Northeastern

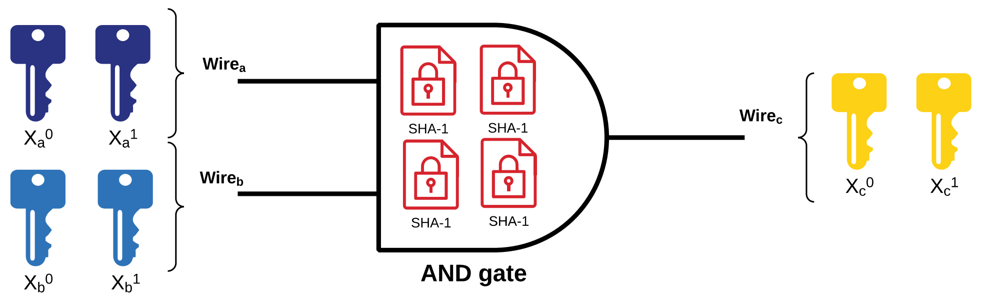
Yao’s Garbled Circuit

- Entities in Yao’s Garbled Circuit Protocol:
  - Users
  - Garbler
  - Evaluator
- Function to be evaluated should be expressed by a Boolean circuit and can then be constructed as a garbled circuit represented as AND and XOR gates
- Garbler generates key pairs to represent bit values 0 and 1 and garbles the circuit
- Evaluator evaluates the circuit and learns the result



function to be evaluated

Garbling an AND gate in Garbled Circuit



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

$\xrightarrow{\text{garbling}}$

a	b	c
X_a^0	X_b^0	X_c^0
X_a^0	X_b^1	X_c^0
X_a^1	X_b^0	X_c^0
X_a^1	X_b^1	X_c^1

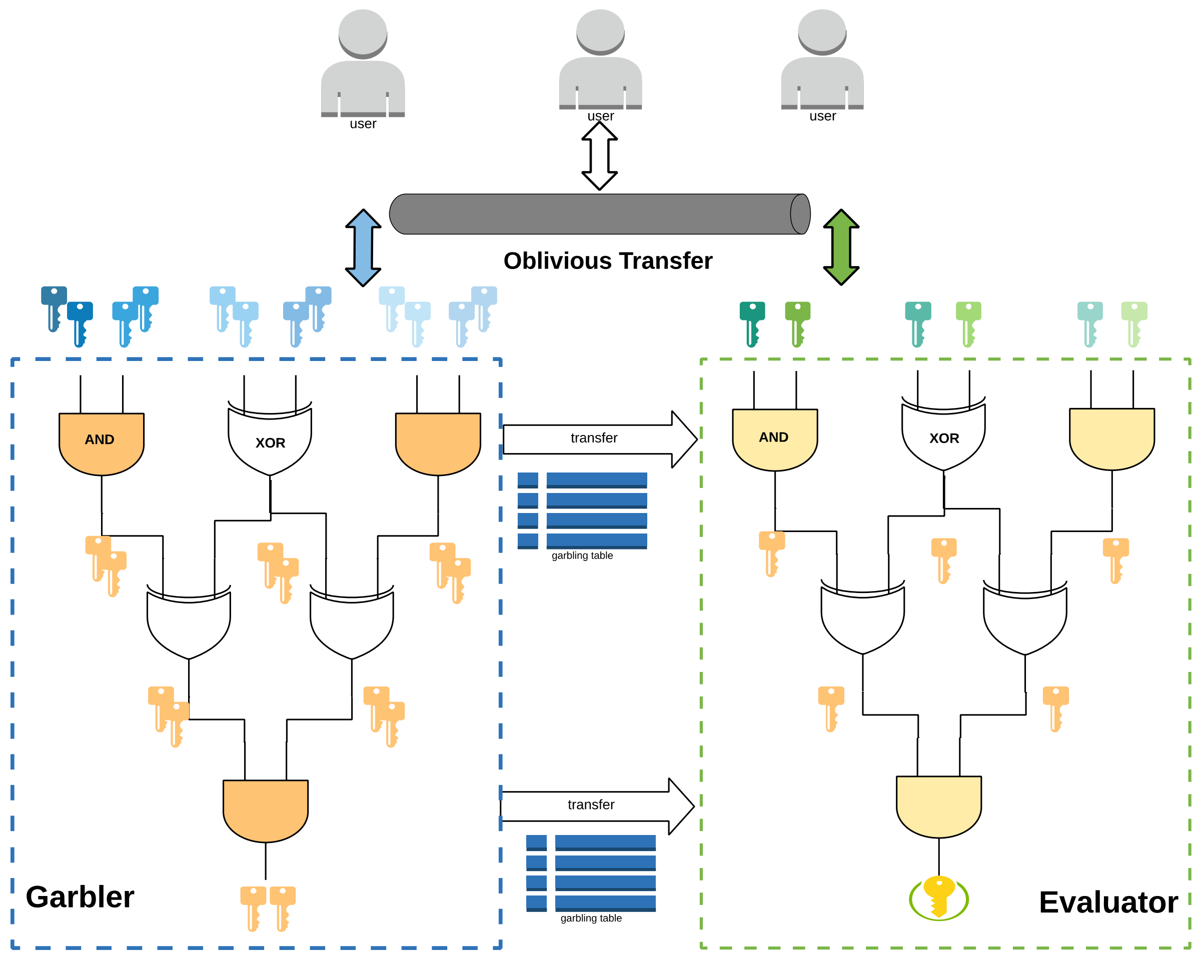
garbling table
$\text{Enc}_{X_a^0\,X_b^0}\ (X_c^0)$
$\text{Enc}_{X_a^0\,X_b^1}\ (X_c^0)$
$\text{Enc}_{X_a^1\,X_b^0}\ (X_c^0)$
$\text{Enc}_{X_a^1\,X_b^1}\ (X_c^1)$

Garbling an AND gate in Garbled Circuit

- AND gate in Garbled Circuit contains 4 SHA-1 cores
- AND gate encrypts the output entry of the truth table and generates the garbling table
- Garbling table needs to be sent to evaluator

Northeastern

Yao’s Garbled Circuit



Garbler and Evaluator in Yao’s Garbled Circuit

- Users, garbler and evaluator engage in proxy oblivious transfer (OT)
- Output keys from the previous gates are used as the inputs of following gates
- Evaluator needs the garbling table from garbler to decrypt the AND gate

Garbled Circuit Optimizations

- Row Reduction [Naor, Pinkas, Summer; EC 1999]

  one ciphertext is picked to be 0

- Point and Permute [Malkhi, Nisan, Pinkas, Sella; USENIX Security 2004]

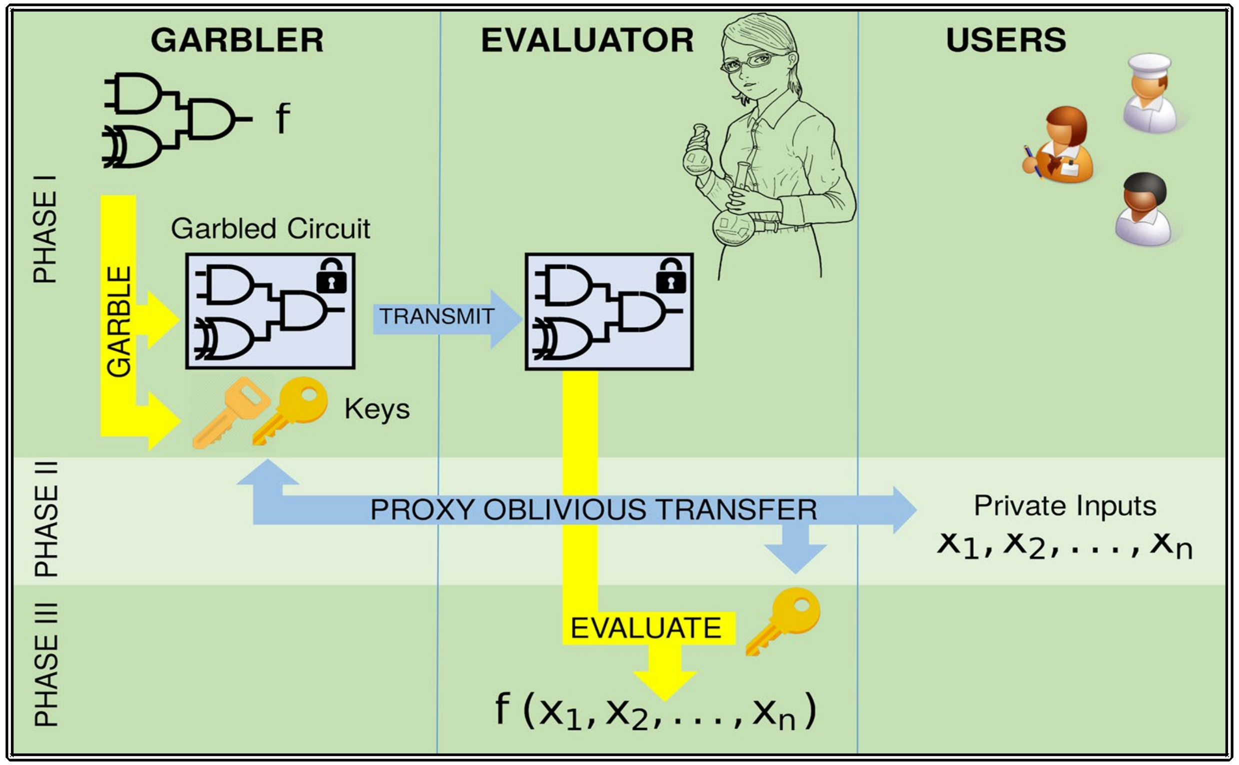
  evaluator needs only decrypt the garbling table once

- Free-XOR [Kolesnikov, Schneider; ICALP 2008]

  output wire keys are calculated by taking XOR of two input keys

Yao’s Garbled Circuit

- Yao’s Garbled Circuit guarantees users’ data privacy
- Garbler facilitates SFE but learns nothing
- Evaluator learns nothing but the output
- The AND gate requires encryption of SHA cores



Garbled Circuit Protocol

Challenges and Contributions

Challenges:

- Garbling significantly slows down function evaluation
- Accelerate any general garbled circuit
- Prove scalability for large datasets

Contributions:

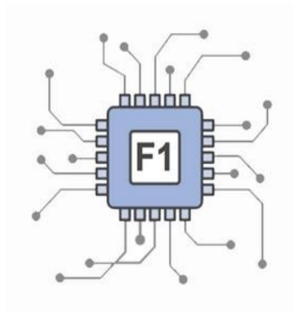
Implemented:

- a hardware FPGA overlay for general garbled circuit problem
- an End-to-End system for garbled circuit in the Cloud
- a complete design on AWS platform

Amazon Web Service (AWS)

AWS Provides:

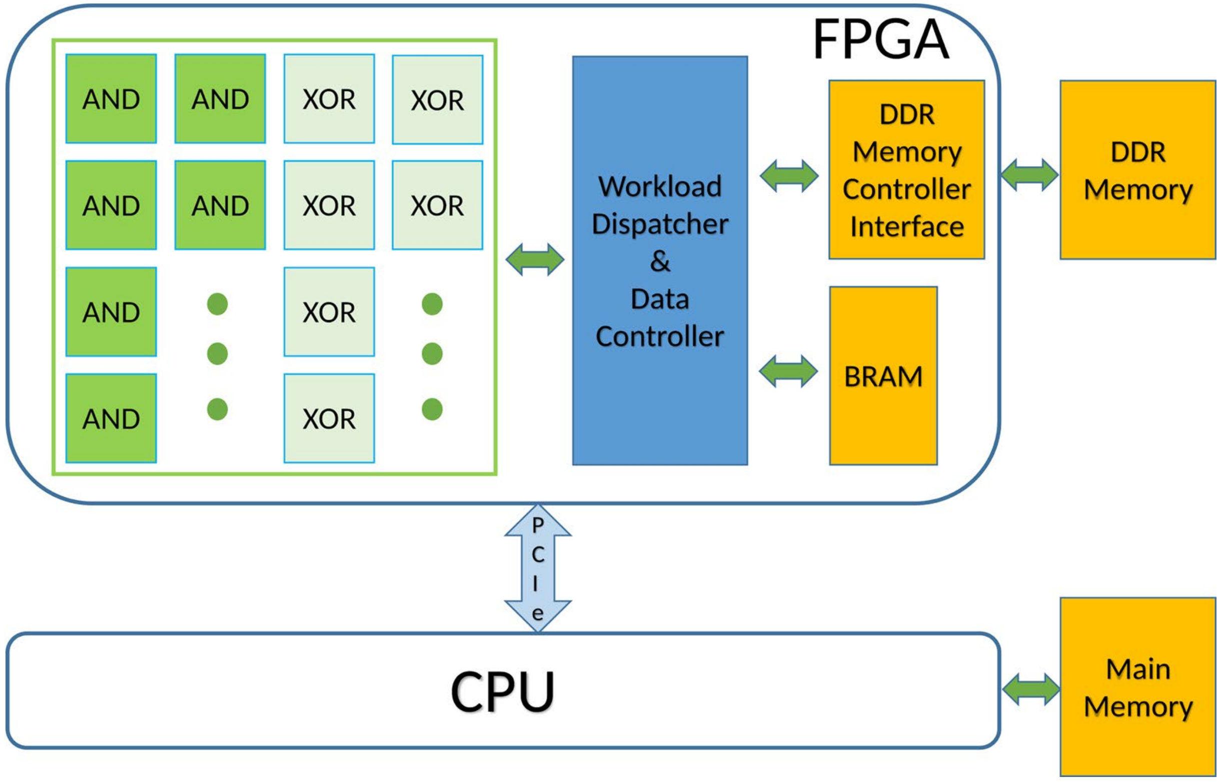
- development environment
- hardware and software development kit
- high-end FPGA boards(UltraScale+ VU9P) on f1 instances



Each Xilinx FPGA includes:

- Local 64 GB DDR4 ECC protected memory
- Dedicated PCIe x16 connections
- Approximately 2.5 million logic elements, 6,800 DSP engines

Coarse-Grained Hardware Overlay

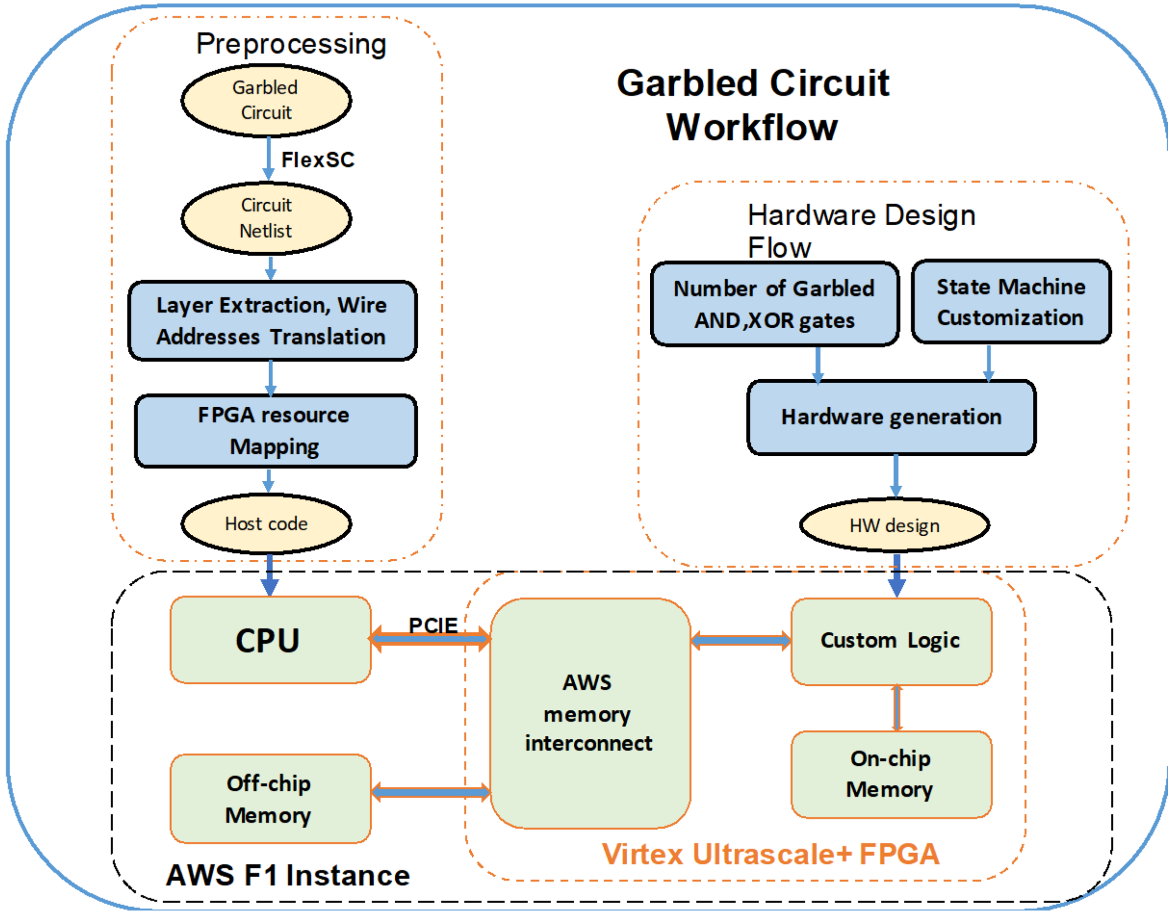


Garbled Circuit Hardware Design

- Needs only be loaded once and used for any garbled circuit problem
- Overlay with different number of AND, XOR gates can be generated
- Coordinates with host C code at runtime

Northeastern

Garbled Circuit workflow

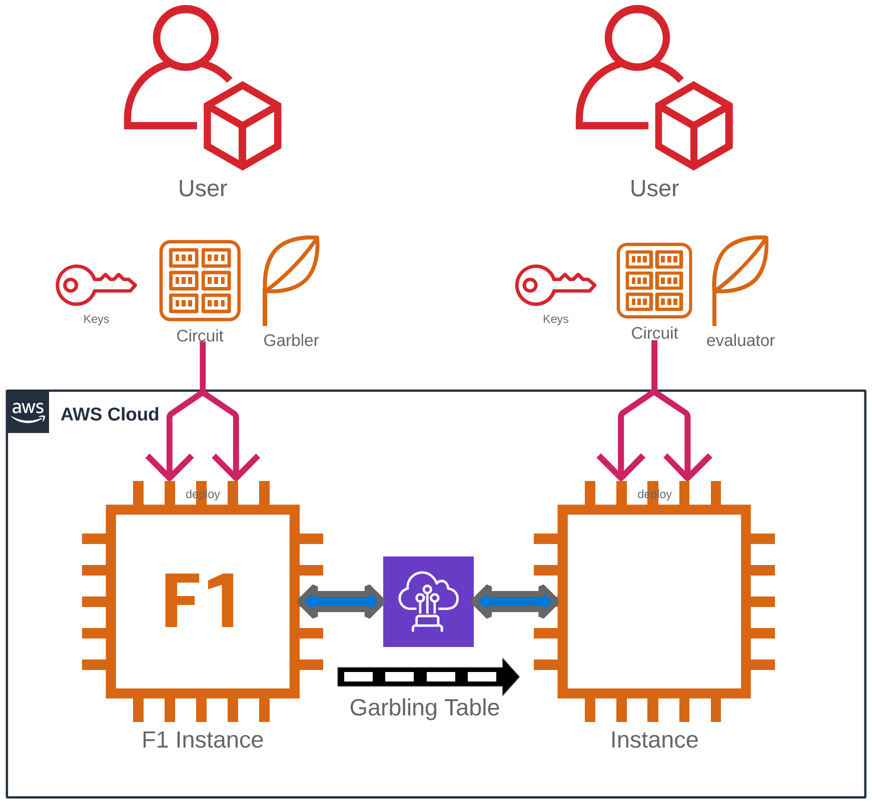


Garbled Circuit Workflow

- Preprocessing extracts layers and translates wire IDs to memory addresses
- Preprocessing partitions the netlist and maps them to FPGA
- Hardware overlay scales according to number of Garbled AND and XOR cores

Northeastern

Experiments



Garbled Circuit Experiments

- The keys are directly generated for the evaluator
- The initial memory layout, FPGA mapping information and runtime addresses are generated for FPGA garbler
- The garbler and evaluator run on two different nodes and the transfer time is estimated by f1 bandwidth
- We record the garbling time and evaluating time

Northeastern

Benchmarks

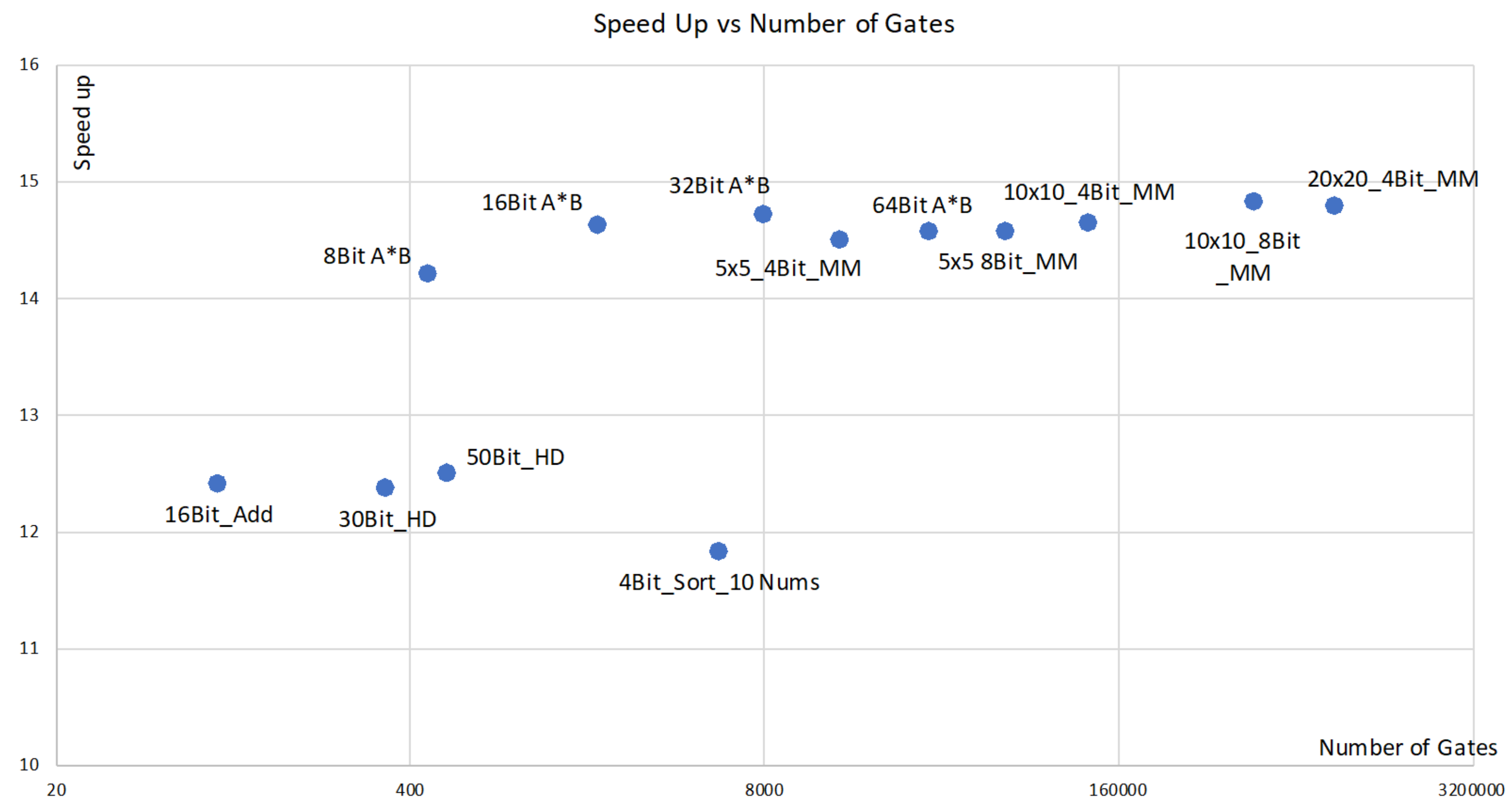
- Size of benchmarks

Problem	Inputs	Outputs	Layers	Gates
16-bit add	32	16	48	80
30-bit HD	60	30	27	330
50-bit HD	100	50	32	550
8-bit multiply	16	8	57	472
16-bit multiply	32	16	121	1968
32-bit multiply	64	32	249	8032
64-bit multiply	128	128	505	32448
10 4-bit sort	40	40	278	5486
5x5 8-bit MM	400	200	57	63000
10x10 4-bit MM	800	400	27	126000
10x10 8-bit MM	1600	800	57	508000
20x20 4-bit MM	3200	1600	37	1016000

HD: Hamming Distance
MM: matrix multiply

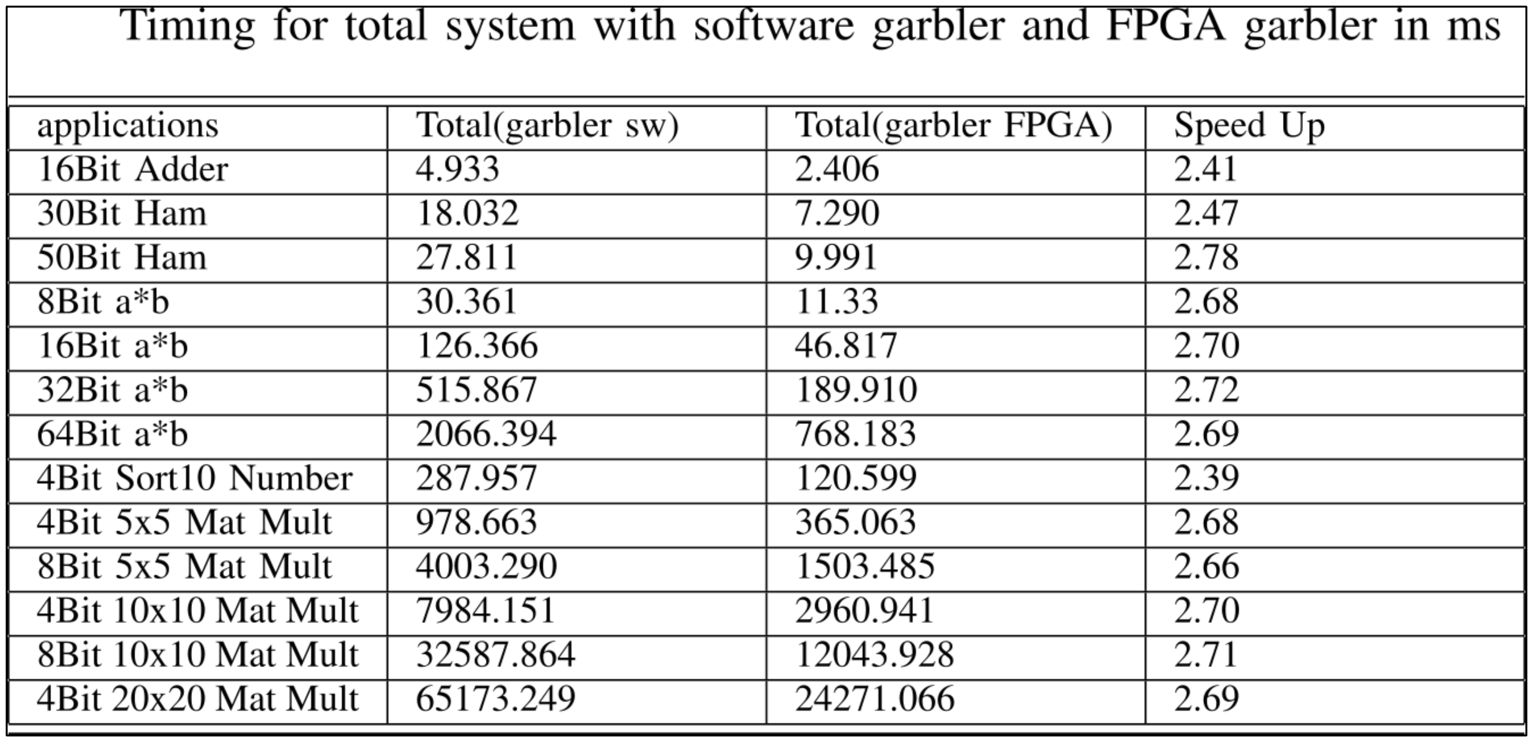
Garbler Timing Speed up

- Garbler Timing Speed Up on AWS



End to end runtime of FPGA garbler and software garbler

- End-to-end runtime system speed up on AWS (unit: ms)

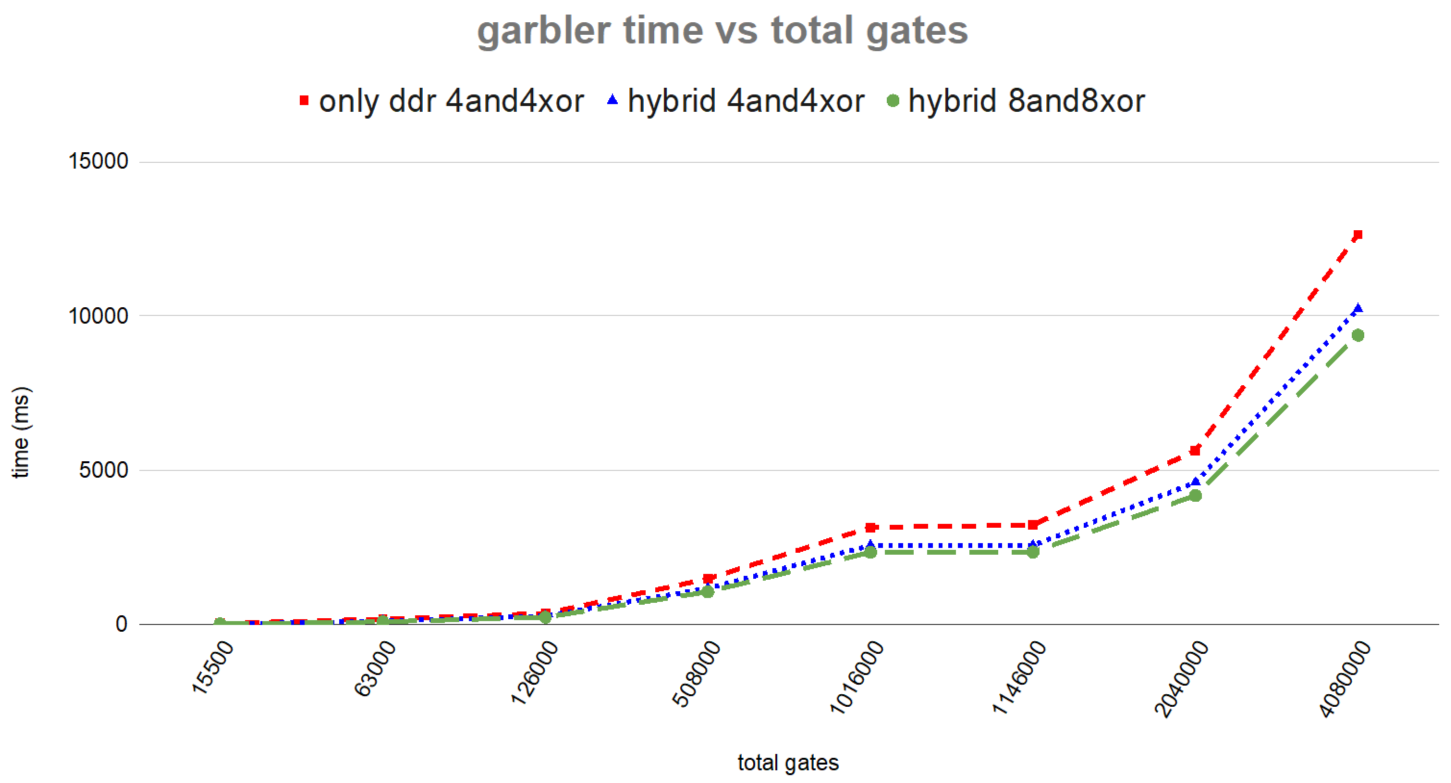


Timing for total system with software garbler and FPGA garbler in ms

applications	Total(garbler sw)	Total(garbler FPGA)	Speed Up
16Bit Adder	4.933	2.406	2.41
30Bit Ham	18.032	7.290	2.47
50Bit Ham	27.811	9.991	2.78
8Bit a*b	30.361	11.33	2.68
16Bit a*b	126.366	46.817	2.70
32Bit a*b	515.867	189.910	2.72
64Bit a*b	2066.394	768.183	2.69
4Bit Sort10 Number	287.957	120.599	2.39
4Bit 5x5 Mat Mult	978.663	365.063	2.68
8Bit 5x5 Mat Mult	4003.290	1503.485	2.66
4Bit 10x10 Mat Mult	7984.151	2960.941	2.70
8Bit 10x10 Mat Mult	32587.864	12043.928	2.71
4Bit 20x20 Mat Mult	65173.249	24271.066	2.69

Garbler timing of different designs

- Garbler with hybrid memory design and different number of cores on AWS (unit: ms)



Hybrid memory design uses both off-chip and on-chip memory

Less is better !

Conclusion and Future Work

Conclusion

- We map Garbled Circuit to FPGA and the hardware design can scale to arbitrary number of AND and XOR cores
- Our garbler gains speed up against software up to **18x** for million gate examples

Future Work

- Replace the SHA-1 cores with AES cores
- Reduce host to FPGA communication
- Map this problem to multiple nodes for big-data processing

Thank you!

email : huang.kai1@husky.neu.edu
https://www.northeastern.edu/rcl/

Thanks to the support of AWS
Thanks to NSF (SaTC1717213)