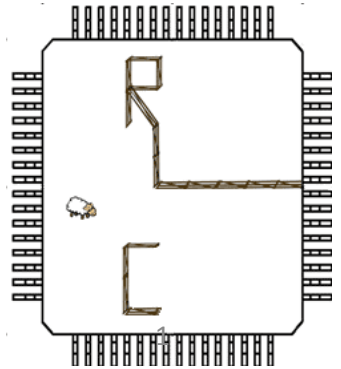
# Secure Function Evaluation using an FPGA Overlay Architecture

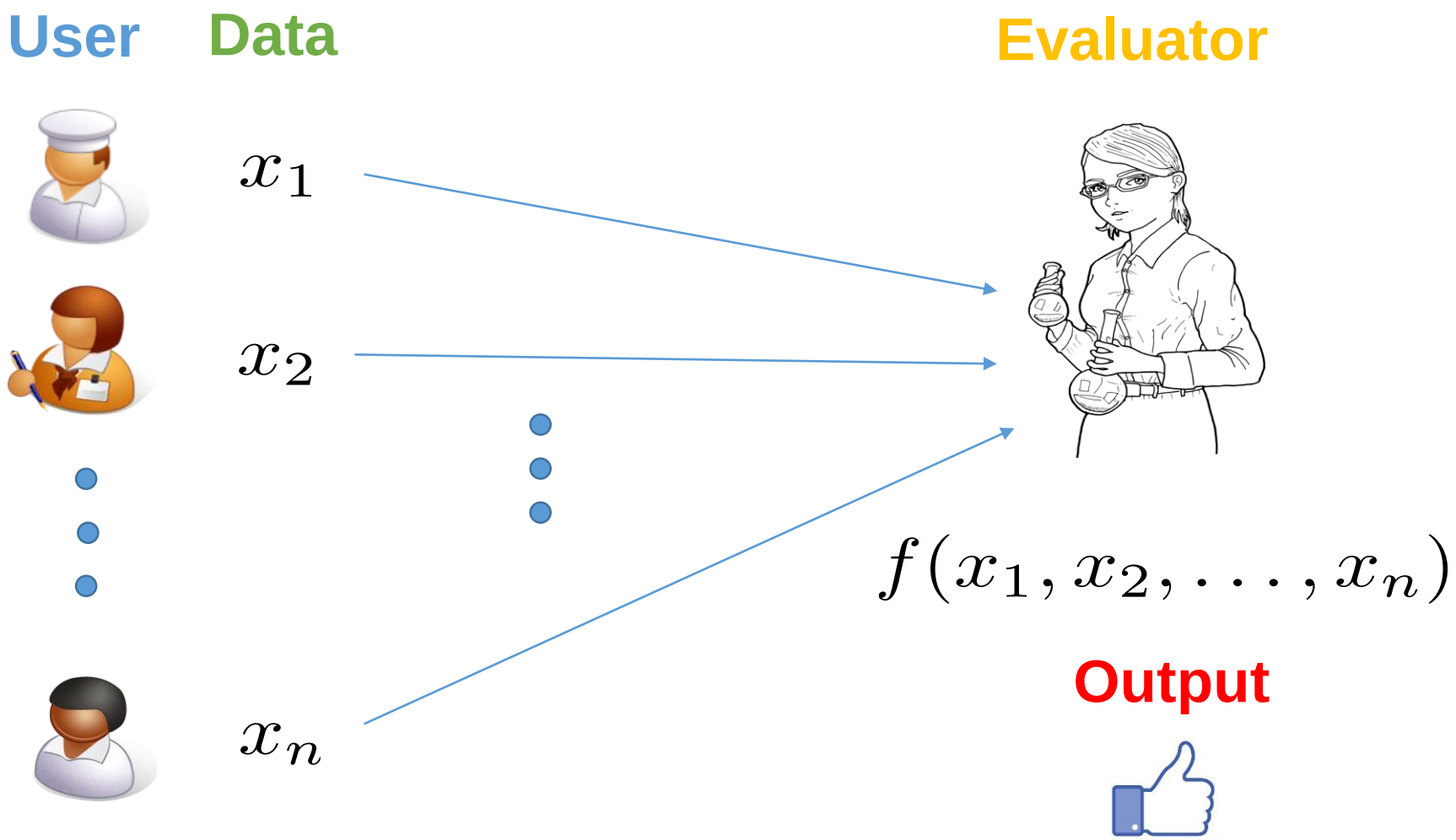
**Xin Fang**
**Stratis Ioannidis**
**Miriam Leeser**

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

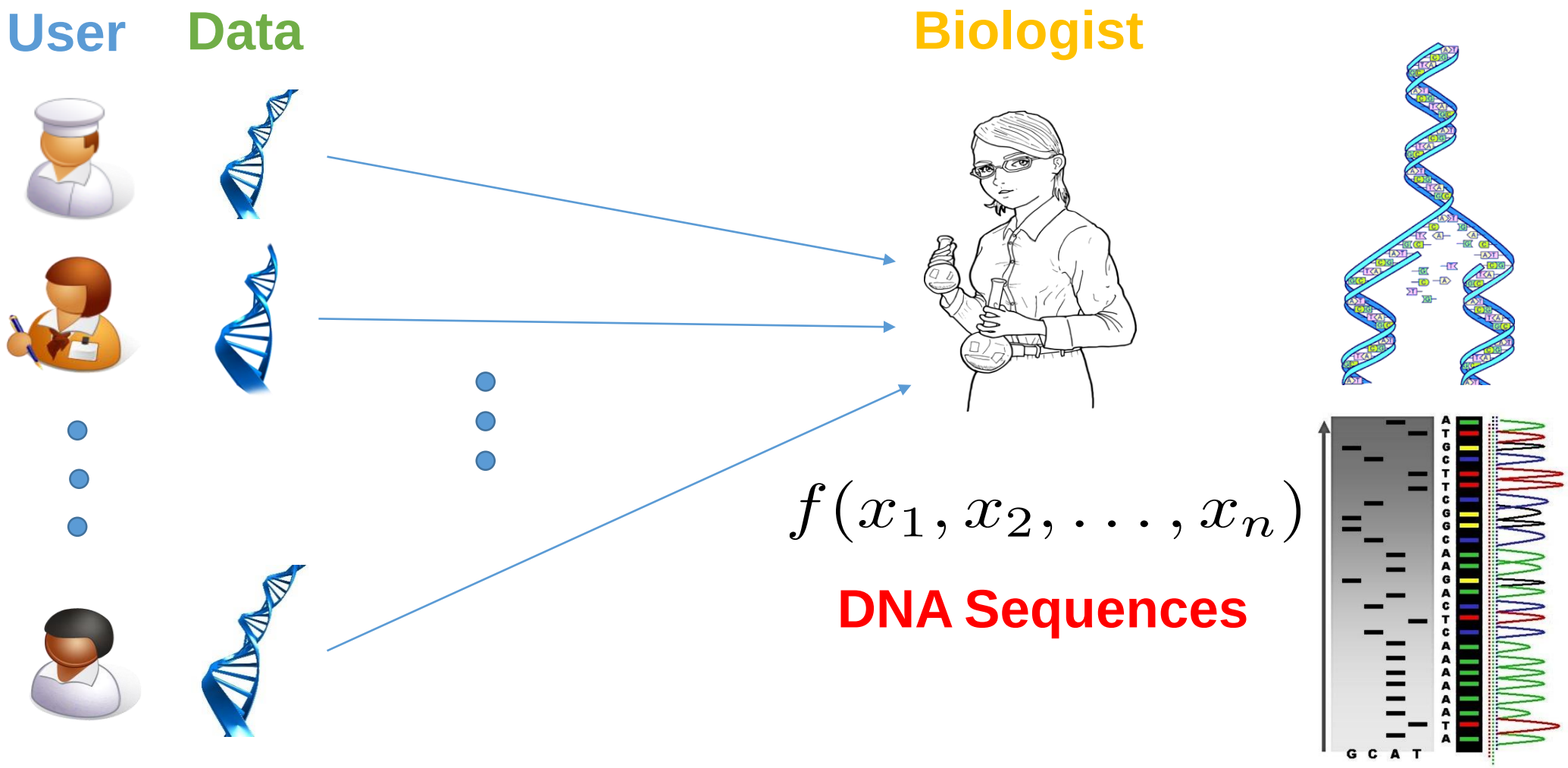



**FPGA 2017**

# Introduction
# **Data Evaluation**



# Examples:

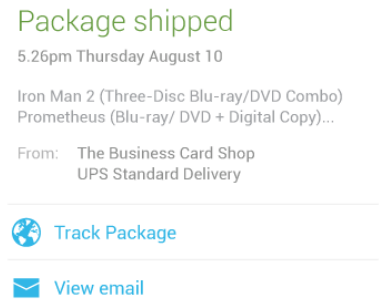


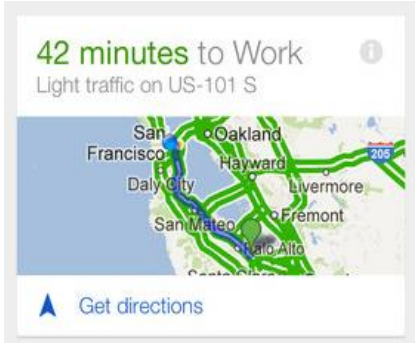
# Examples:

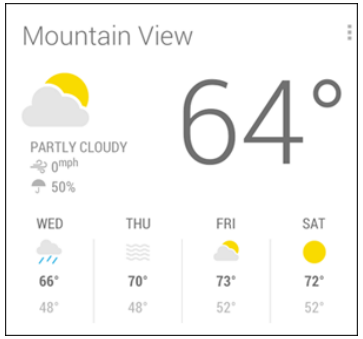
## User

## Internet Footprint

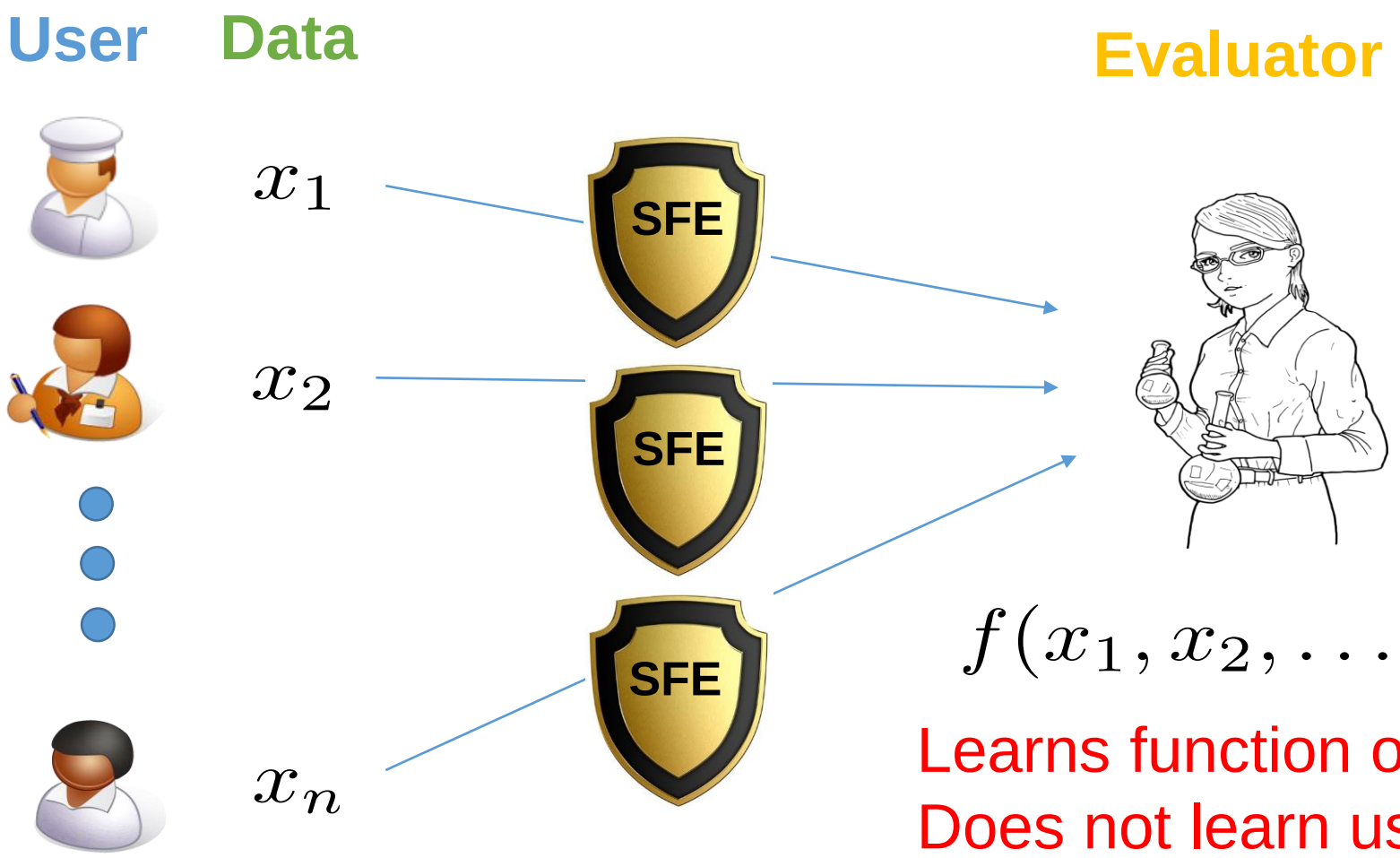




# Data is protected using Secure Function Evaluation (SFE)




Learns function output **only**;
Does not learn user data

# Garbled Circuits (GC)

- One solution of SFE
- Introduced by Andrew Yao in 1982
- Use **cryptographic primitives**
- Suitable for any function represented as a Boolean circuit
- Three types of parties:
  - Garbler, Evaluator, Users

# Current Problems and Situation

- **Software Platform** Limitations :
  - Slow on large problems
  - Energy inefficient
- **Acceleration** is needed
- Main objective: Accelerate Garbled Circuits using FPGA architecture

# Traditional FPGA Workflow for GC Problems

- Disadvantages:
  - GC problems too large to map
  - Long compilation time
  - Redo the implementation for every new problem

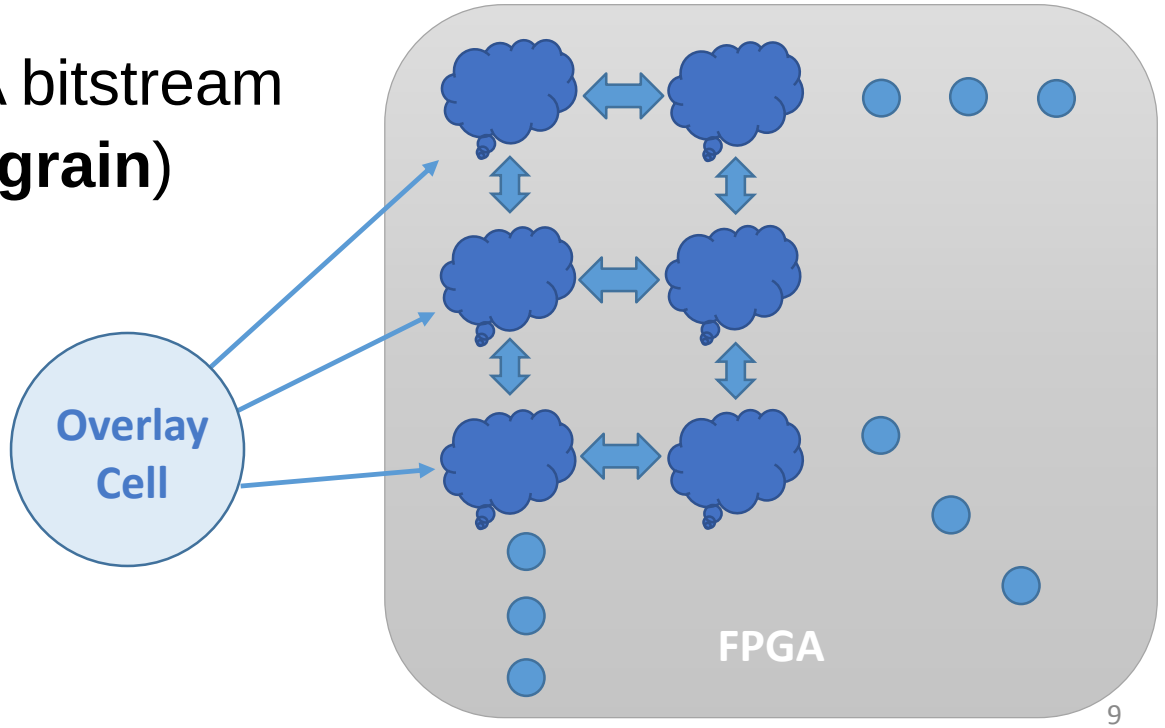
- Solution:

  - **FPGA Coarse Grain Overlays**

# FPGA Overlays

- **Two parts**:
  - A circuit design implemented on FPGA fabric using standard design flow
  - A user circuit mapped onto that overlay circuit

- **Benefits**:
  - User-configured logic in fixed FPGA bitstream
  - Optimized for GC problem (**coarse grain**)
  - One time synthesis



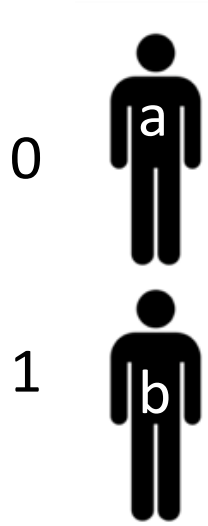
# Our Contributions

- **First** FPGA overlay architecture to accelerate generic Garbled Circuit problems
- **Tools and workflow** for modeling and mapping GC problems onto FPGAs
- Show speedup of **2-3 orders of magnitude**

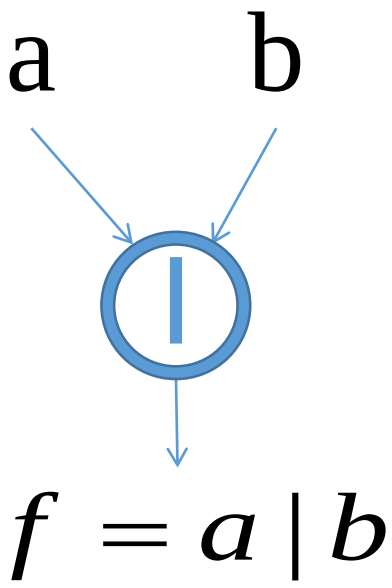
# Garbled Circuits - **Steps**

Evaluator and two users with private data

## Users




## Evaluator

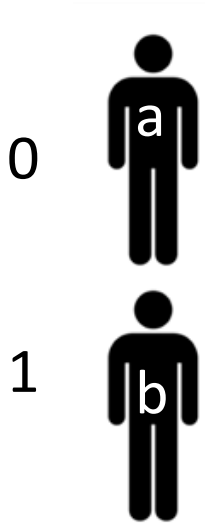


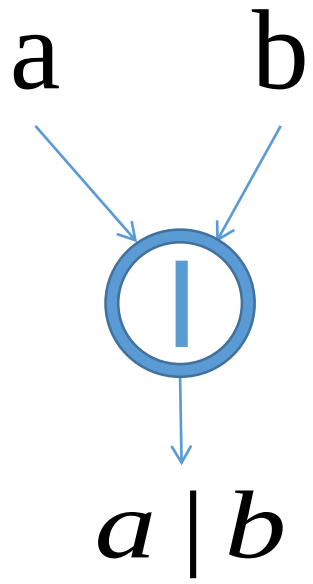

| Inputs | | Outputs |
|---|---|---|
| a | b | a or b |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

# Garbled Circuits - **Steps**

**Garbler:** encrypt the outputs, shared to Evaluator

**Users** **Evaluator** **Garbler**



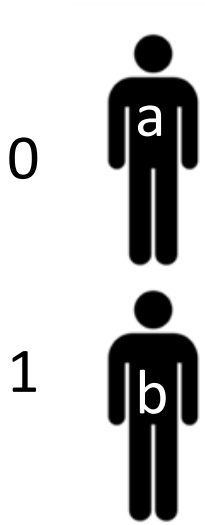



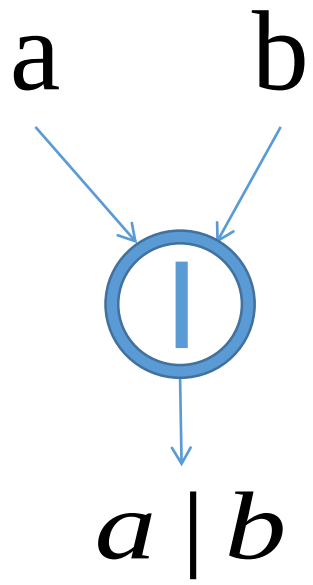

| Inputs | | Outputs |
|---|---|---|
| a | b | x |
| $a_0$ | $b_0$ | $Enc_{a0,b0}(x_0)$ |
| $a_0$ | $b_1$ | $Enc_{a0,b1}(x_1)$ |
| $a_1$ | $b_0$ | $Enc_{a1,b0}(x_1)$ |
| $a_1$ | $b_1$ | $Enc_{a1,b1}(x_1)$ |

# Garbled Circuits - **Steps**

**Garbler:** encrypt the outputs, shared to Evaluator

**Users** **Evaluator** **Garbler**



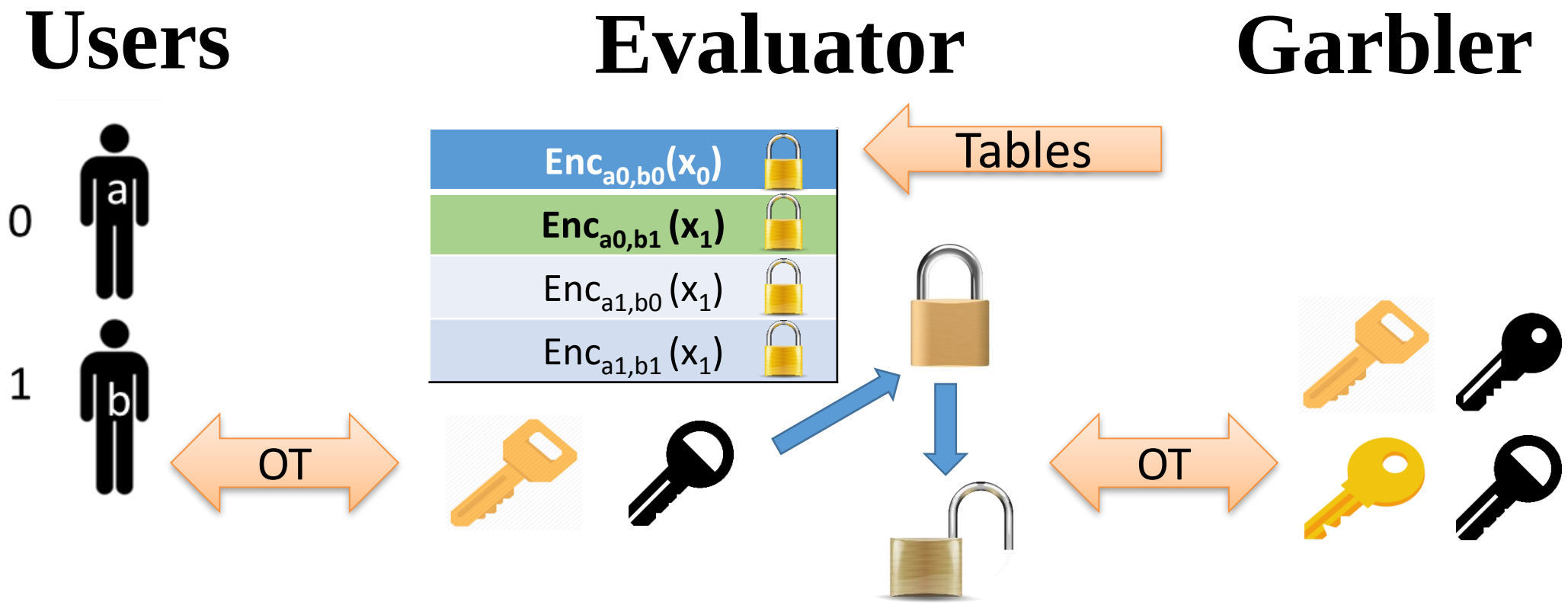




| Inputs | | Outputs |
|---|---|---|
| a | b | x |
| $a_0$ | $b_0$ | $Enc_{a0,b0}(x_0)$ |
| $a_0$ | $b_1$ | $Enc_{a0,b1}(x_1)$ |
| $a_1$ | $b_0$ | $Enc_{a1,b0}(x_1)$ |
| $a_1$ | $b_1$ | $Enc_{a1,b1}(x_1)$ |

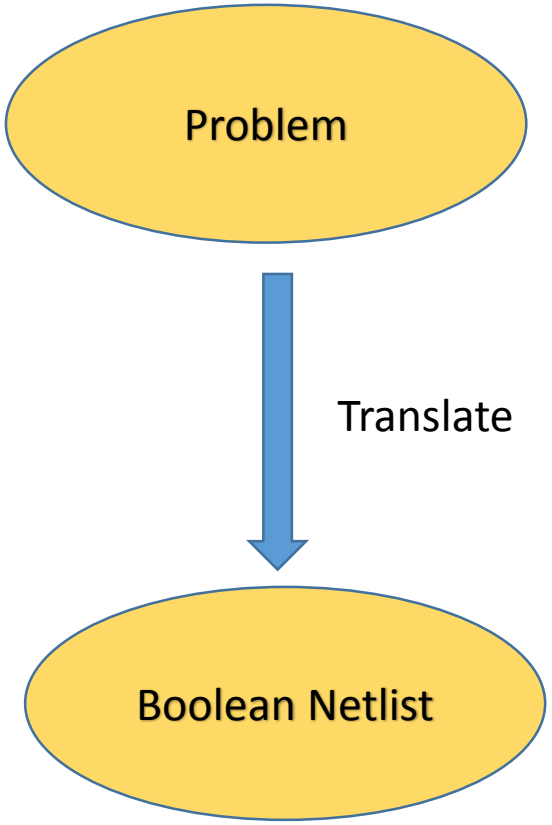
# Garbled Circuits - **Steps**

Using **Oblivious Transfer (OT)**, evaluator

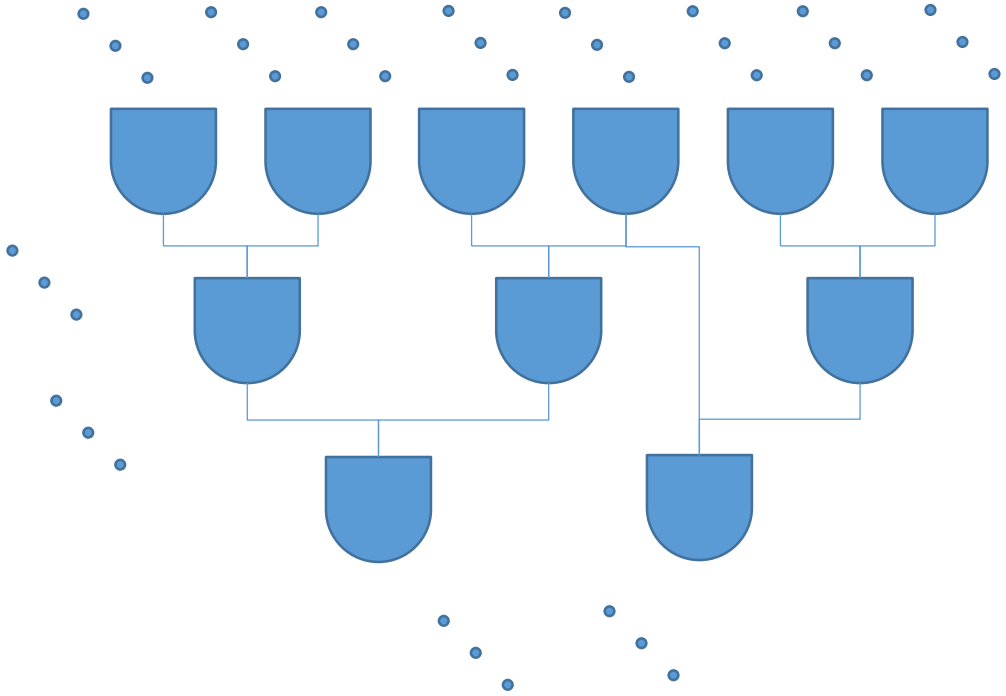
1) Learns **garbled inputs**
2) Evaluates function by decrypting only the matching entry



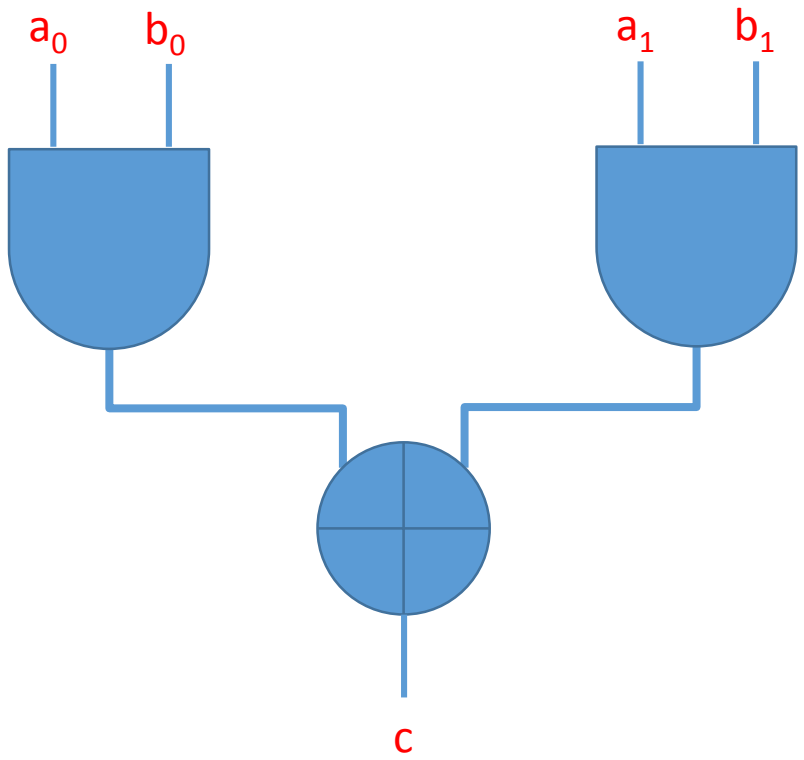
# General Case



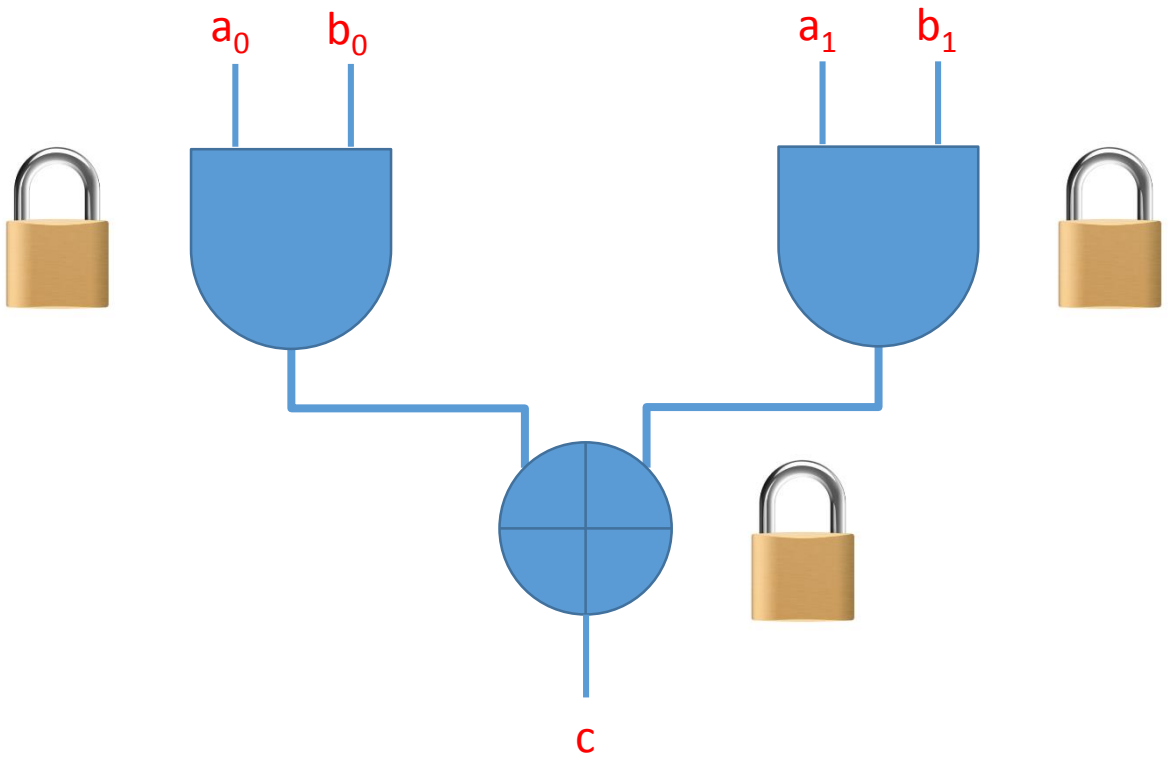

Garbled circuits: suitable for any program that terminates in finite time



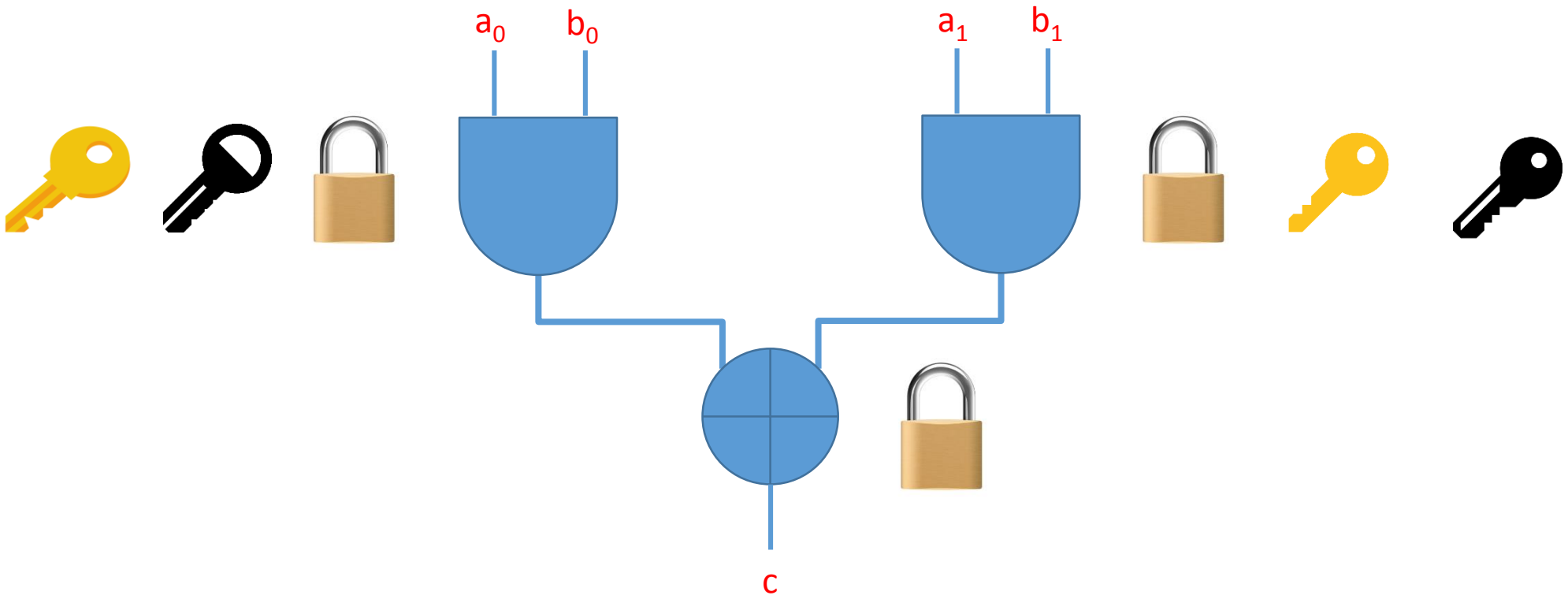
# General Case



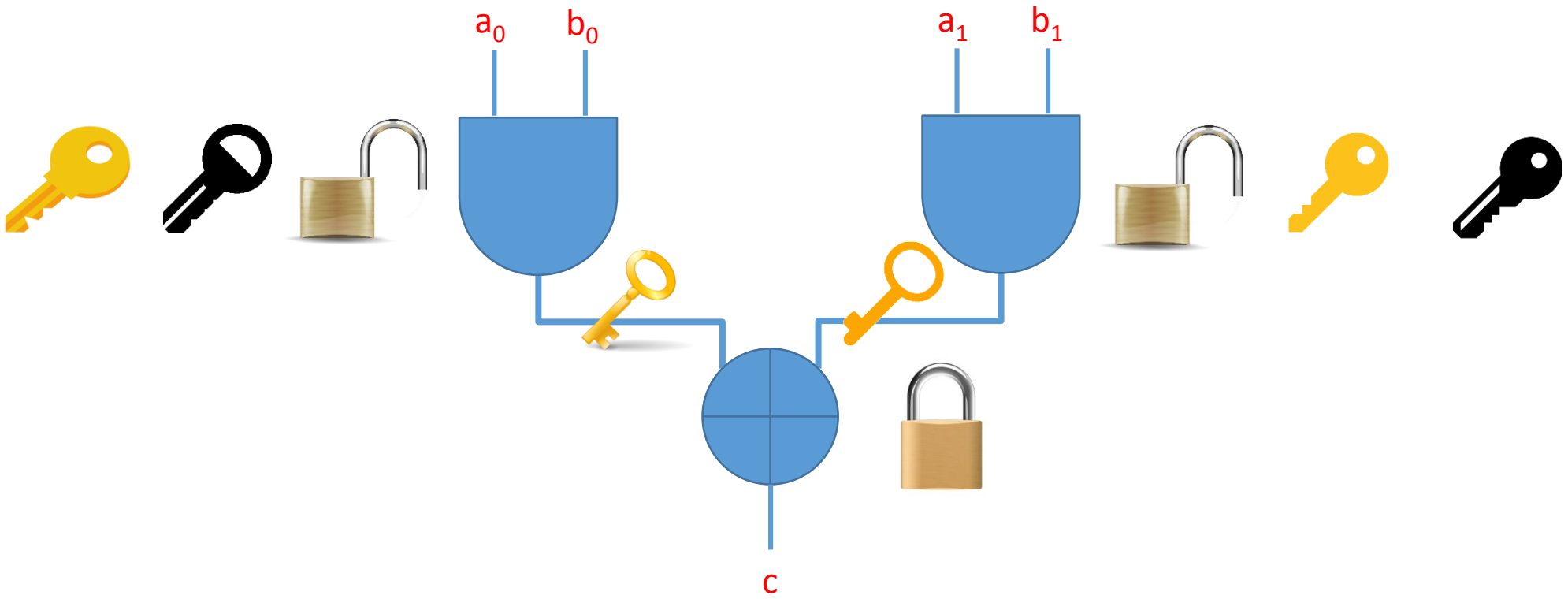
# General Case



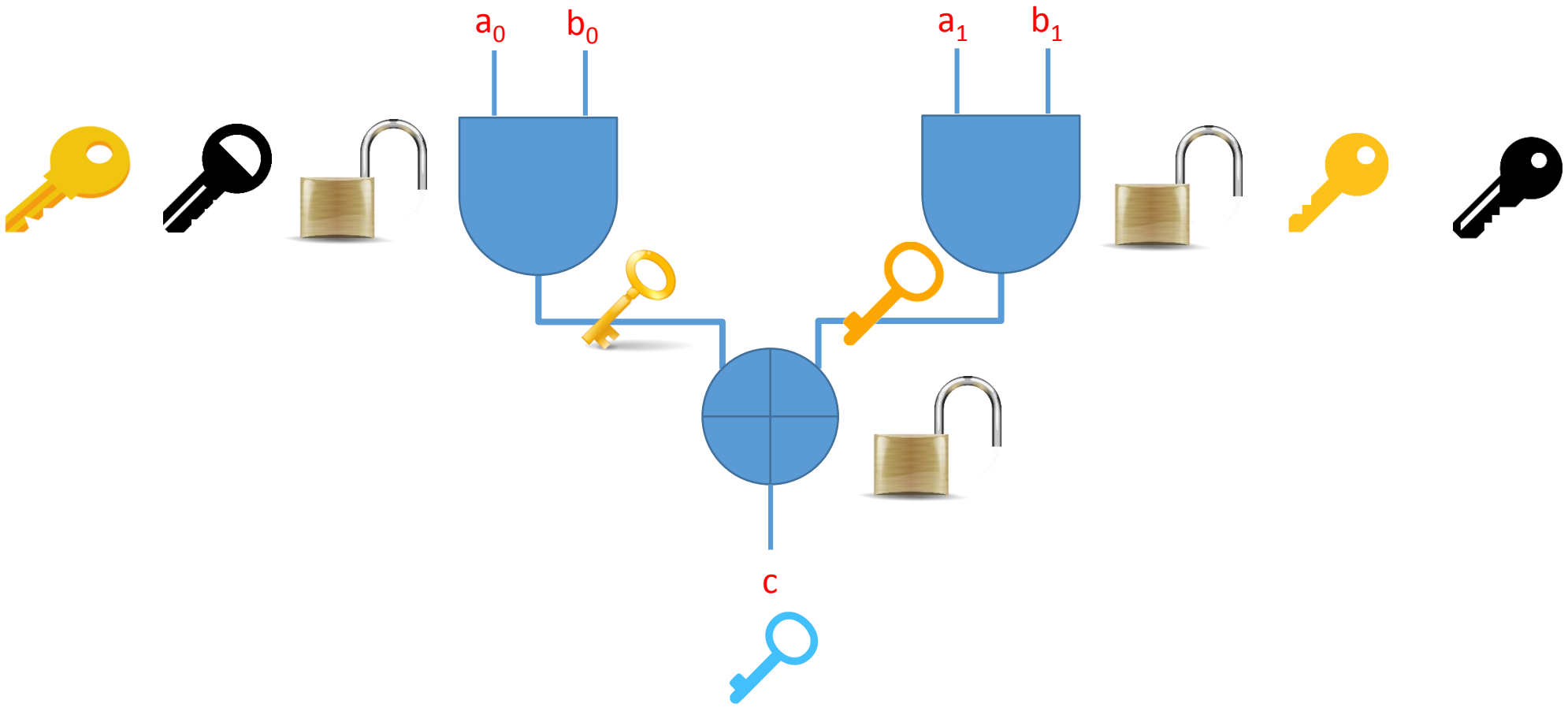
# General Case



# General Case



# General Case



# Takeaway Message

- GC ensures the privacy of input data while being processed
- Garbled Gate is not simple gate
- Arbitrary function that terminates in finite time can be represented as Garbled Circuits

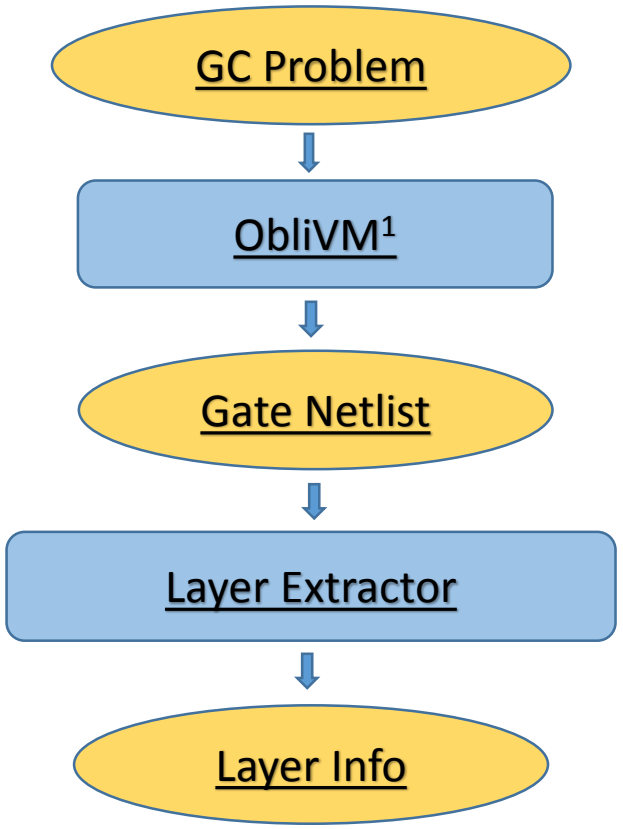
# Solution
# **Challenges**

**Challenge1** Garbled Circuits Acceleration Architecture Design on FPGAs

**Challenge2** Circuits too large to map directly to FPGAs -> How to represent large circuits on FPGAs

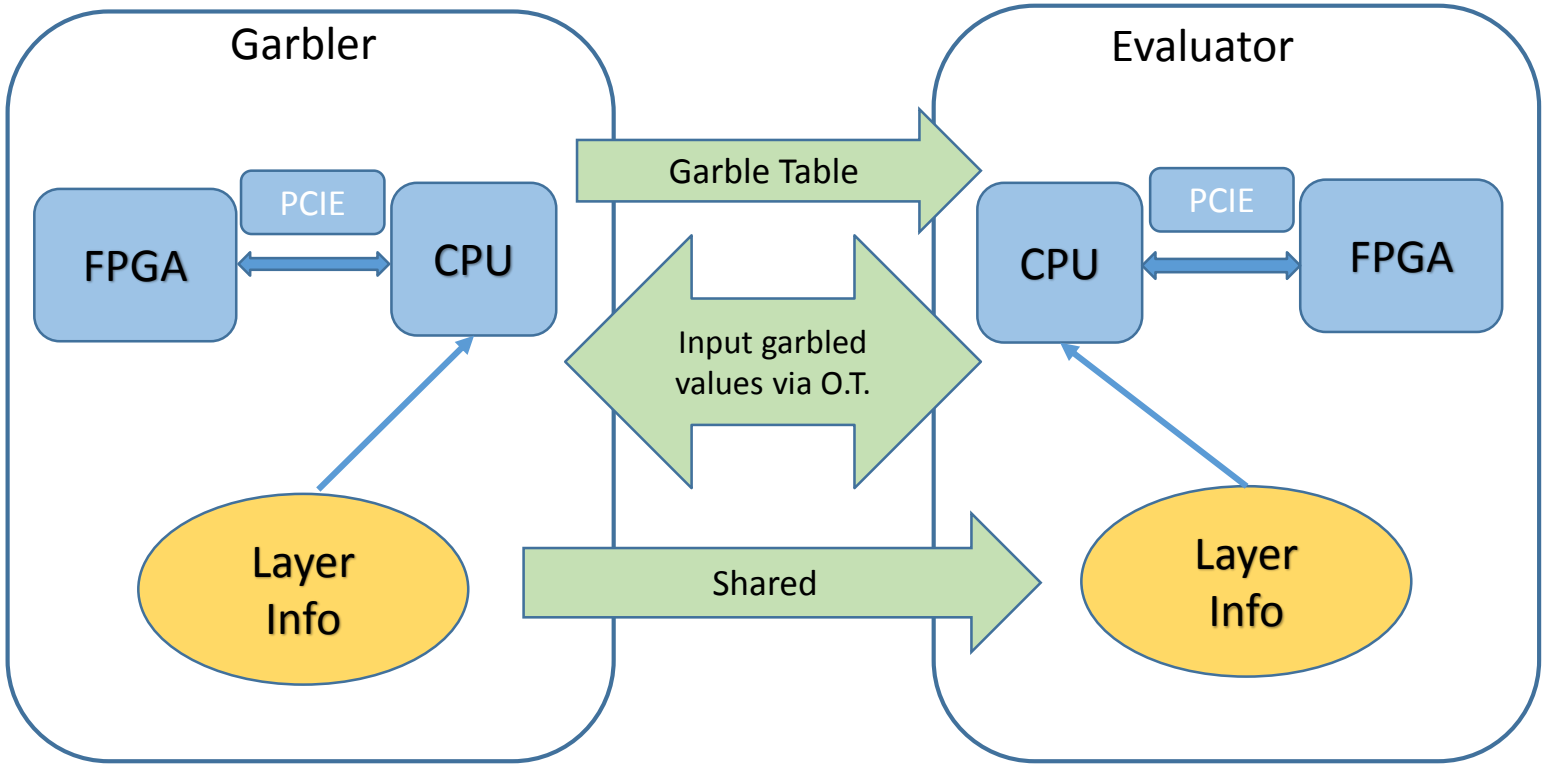
**Challenge3** FPGA and Host CPU Communication -> How to handle data transmission

# Generating Garbled Circuits

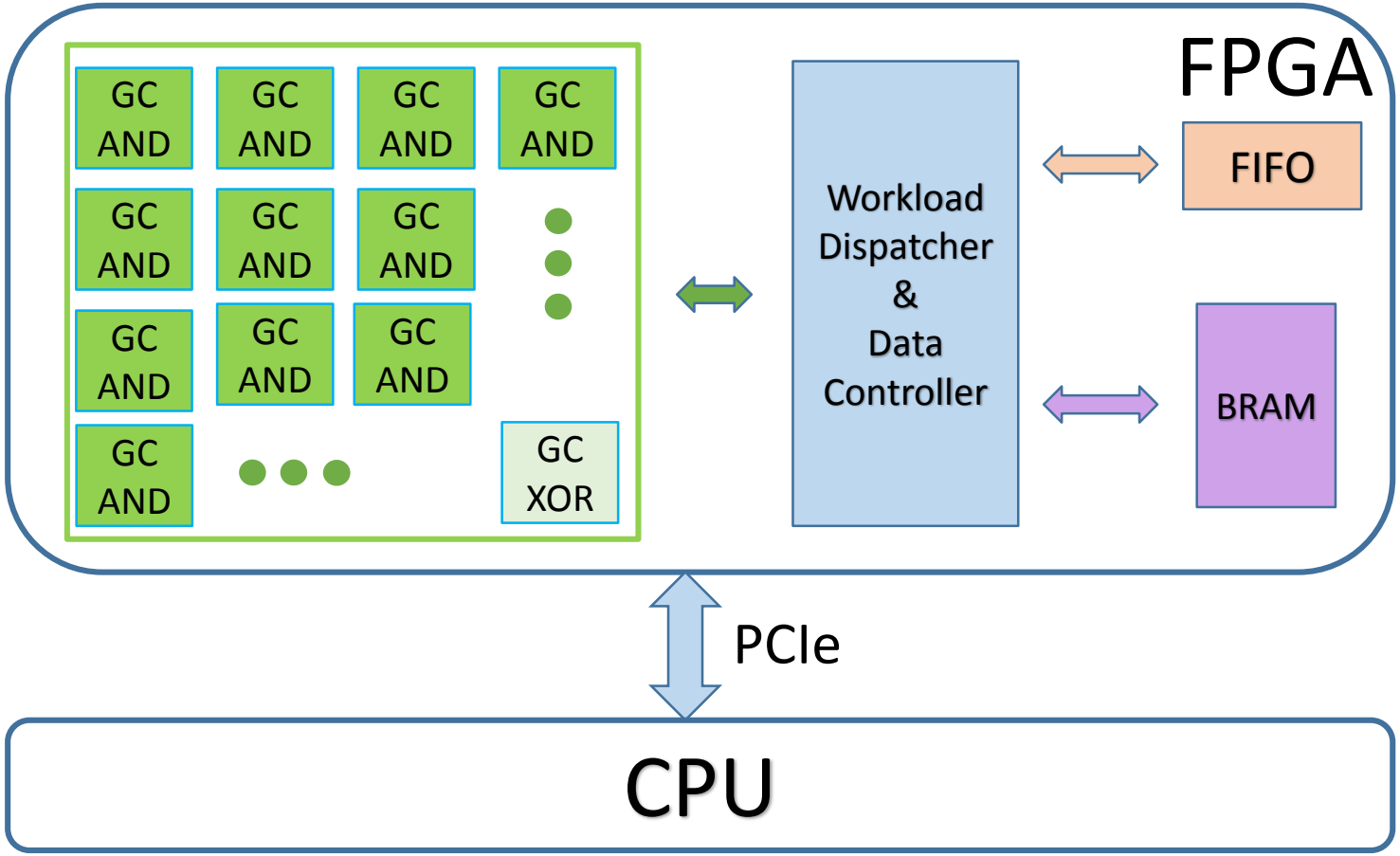



[1] Chang Liu, Xiao Shaun Wang, Karthik Nayak, Yan Huang, and Elaine Shi. ObliVM: A generic, customizable, and reusable secure computation architecture. In IEEE S & P, 2015. http://oblivm.com/

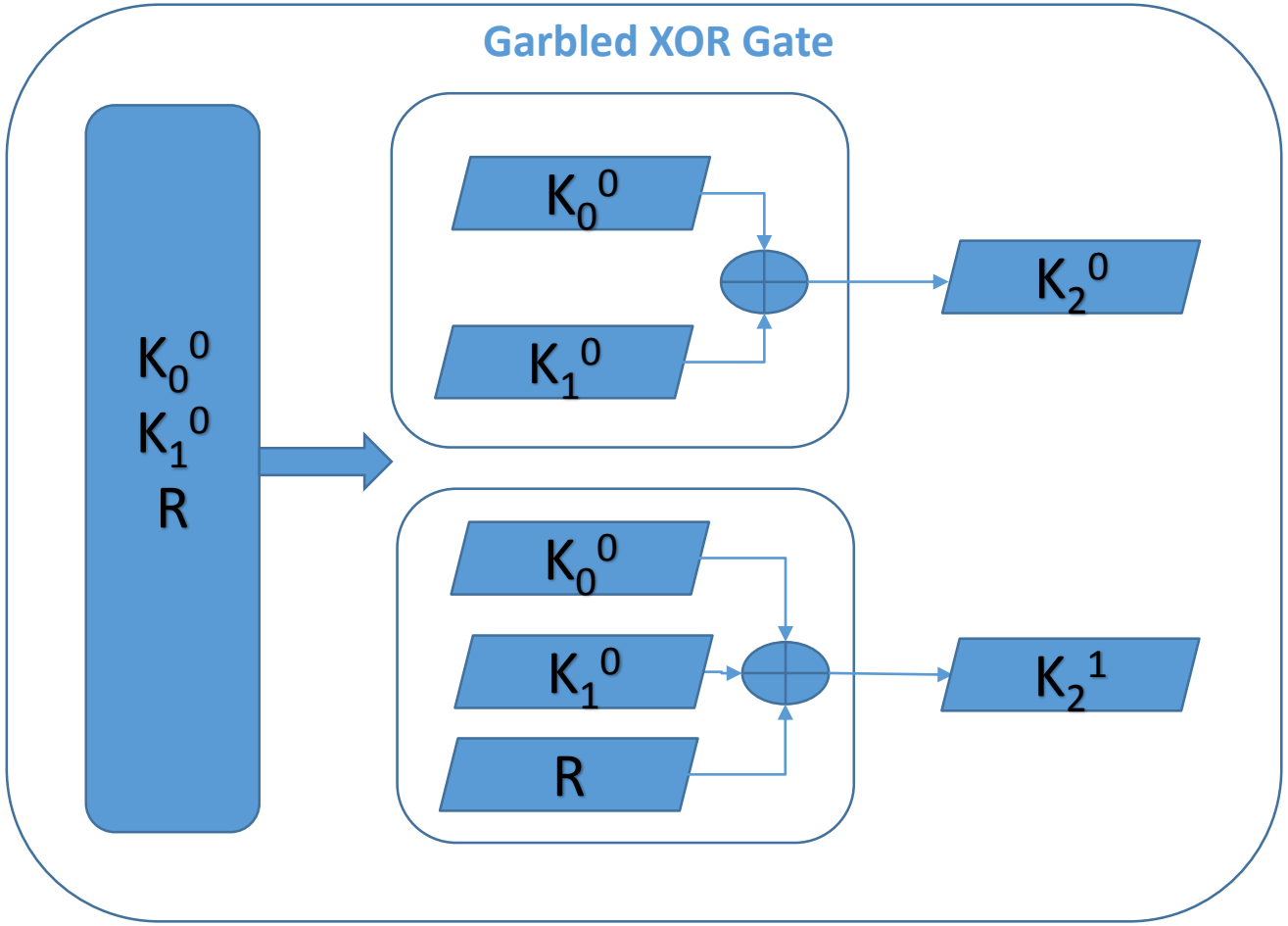
# Garbler and Evaluator on FPGAs



# Garbler Overlay Architecture



# Garbled XOR Gate

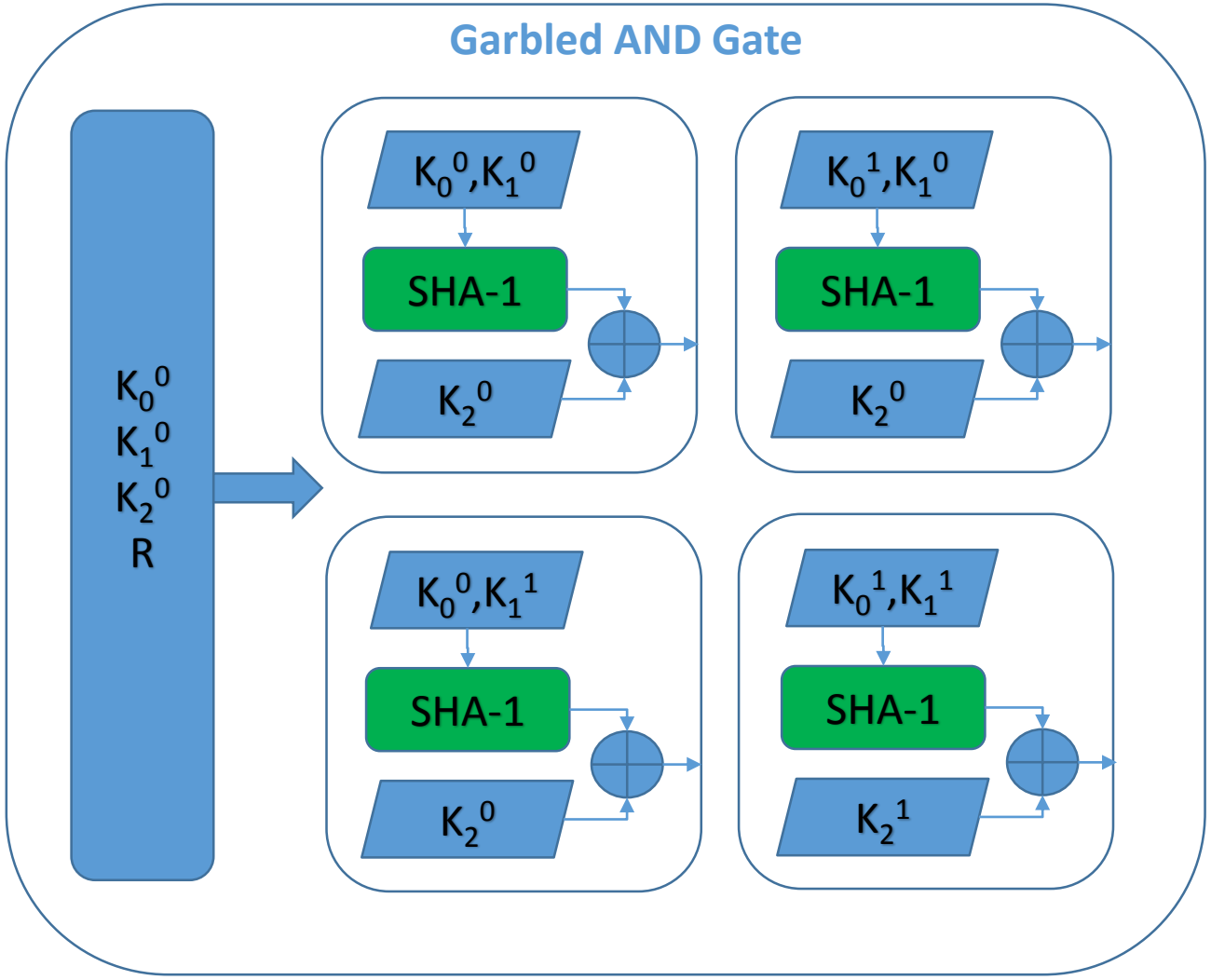



$$K_i^{true} = K_i^{false} \text{ xor } R$$

Latency:
1 cycle

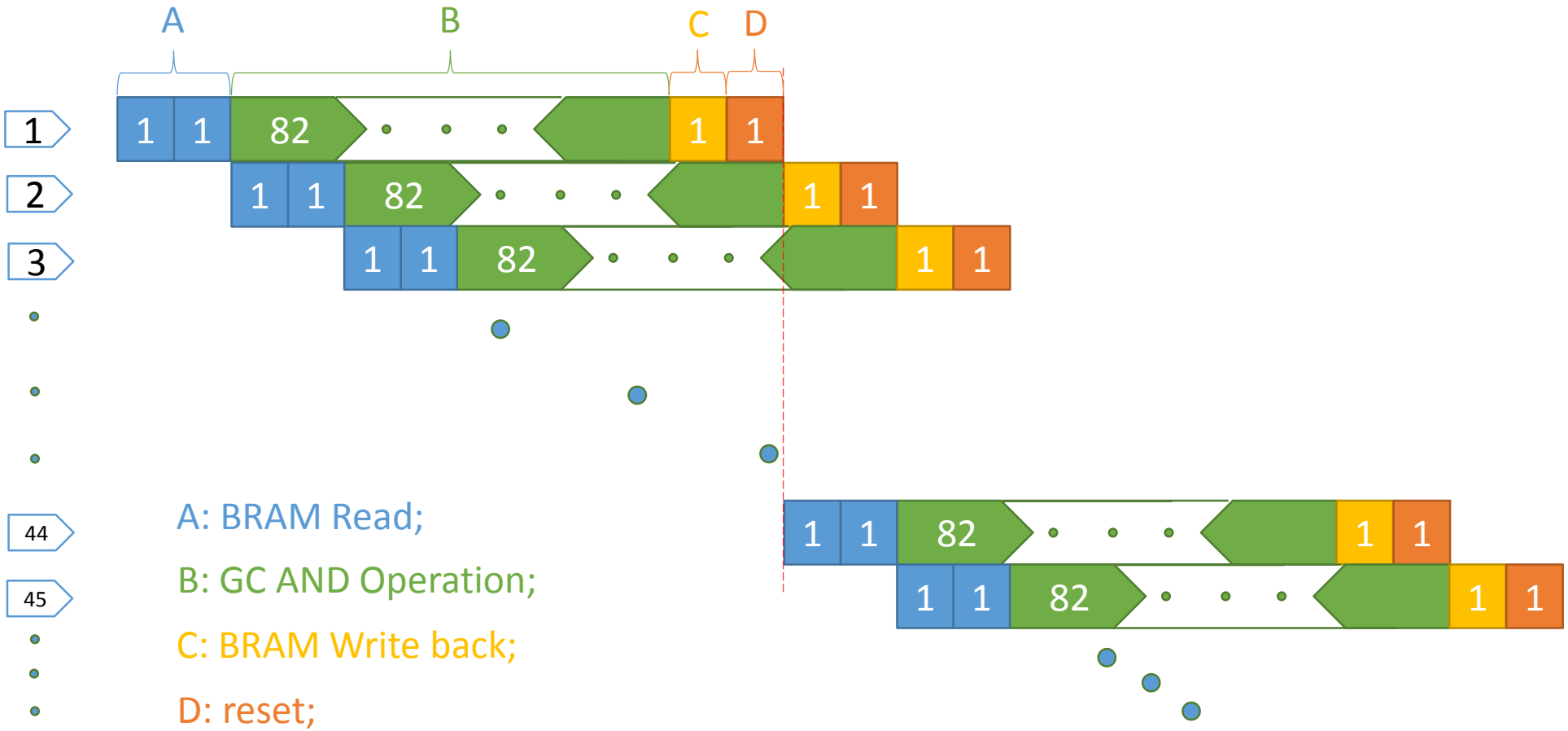
Kolesnikov, Vladimir, and Thomas Schneider. "Improved garbled circuit: Free XOR gates and applications." *International Colloquium on Automata, Languages, and Programming*. 2008.

# Garbled AND Gate




Latency:
82 cycles

# Timing on the FPGA



# Size of Problem

| Problem | # AND Gate | # XOR gate | # layers | Max # AND in One Layer |
|---|---|---|---|---|
| Millionaire(2) | 2 | 11 | 7 | 1 |
| Addition(6) | 6 | 24 | 18 | 1 |
| HD(10) | 20 | 90 | 22 | 5 |
| HD(30) | 60 | 270 | 28 | 15 |
| HD(50) | 100 | 450 | 33 | 25 |
| A*B(8) | 120 | 352 | 57 | 64 |
| A*B(32) | 2016 | 6016 | 250 | 1024 |
| A*B(64) | 8128 | 24320 | 505 | 4096 |
| Sorting(10*4) | 848 | 4683 | 278 | 32 |

# Run times: FPGA Overlay vs. ObliVM

| Problem | Our Approach (Clock cycle) | ObliVM (Clock Cycle) |
|---|---|---|
| Millionaire(2) | $1.9*10^2$ | $1.1*10^6$ |
| Addition(6) | $5.6*10^2$ | $1.7*10^6$ |
| HD(10) | $1.2*10^3$ | $4*10^6$ |
| HD(30) | $2.2*10^3$ | $1.1*10^7$ |
| HD(50) | $2.8*10^3$ | $1.7*10^7$ |
| A*B(8) | $4.4*10^3$ | $3*10^7$ |
| A*B(32) | $3.6*10^4$ | $1.1*10^8$ |
| A*B(64) | $1.1*10^5$ | $3.2*10^8$ |
| Sorting(10*4) | $1.1*10^4$ | $1.4*10^8$ |

# Speedup: FPGA Overlay vs. ObliVM

| Problem | Speedup |
|---|---|
| Millionaire(2) | 422 |
| Addition(6) | 222 |
| HD(10) | 243 |
| HD(30) | 357 |
| HD(50) | 434 |
| A*B(8) | 498 |
| A*B(32) | 218 |
| A*B(64) | 208 |
| Sorting(10*4) | 929 |

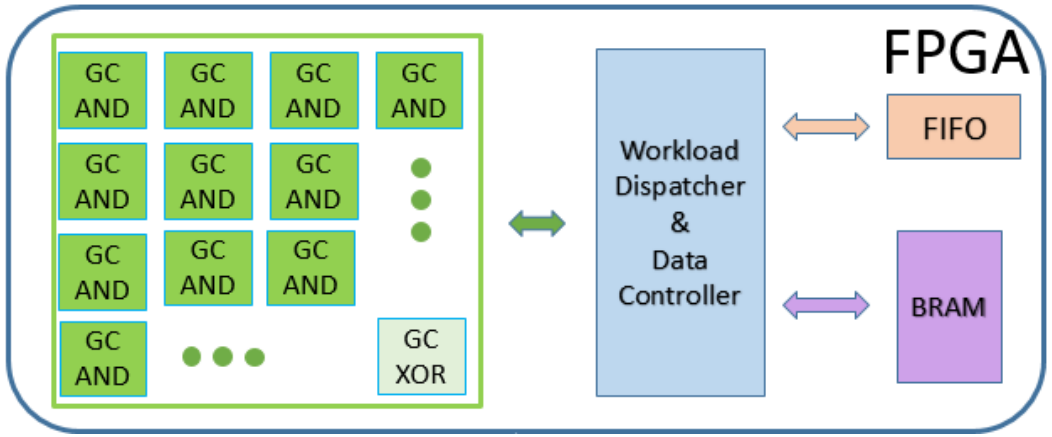
ObliVM runs at 2.80 GHz
FPGA overlay runs at 200 MHz

# Hardware Platform


**Gidel ProceV Board featuring Stratix V FPGA**

# Resource Utilization

| Module | ALMs | M20K |
|---|---|---|
| One And | 3,070 | 0 |
| One XOR | 40 | 0 |
| BRAM | 0 | 1,060 |
| FIFO | 510 | 280 |
| Whole Design | 176,893/234,720 (75.4%) | 1,340/2,560 (52.3%) |

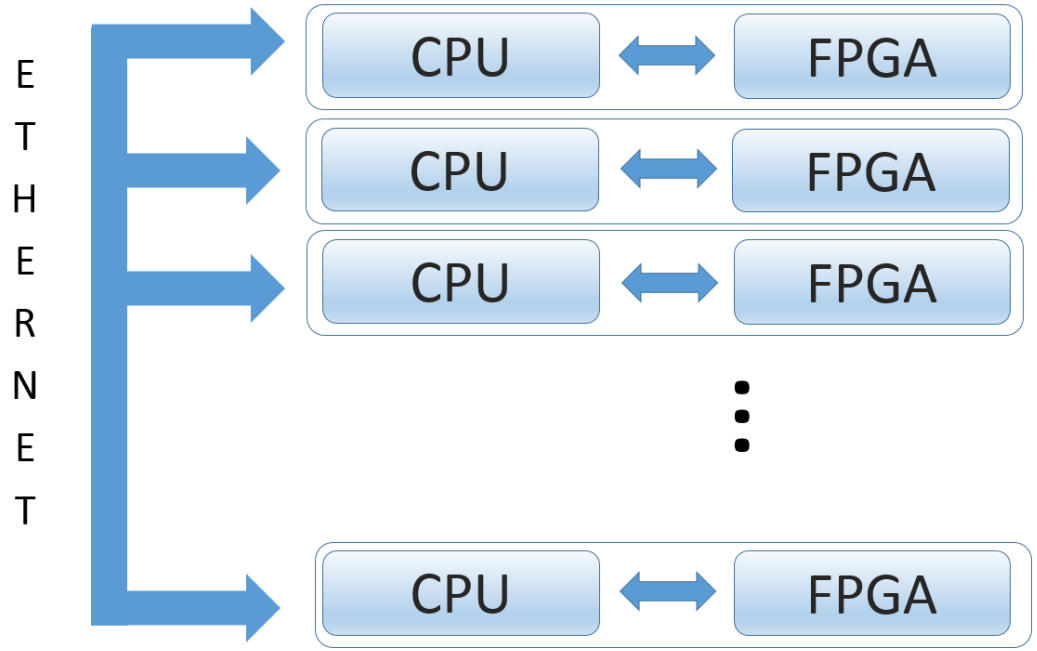


# Conclusions

- First FPGA overlay architecture to accelerate generic Garbled Circuit problems
- Tools and workflow for modeling and mapping GC problems onto FPGAs
- FPGAs are a good platform for Garbled Circuit problem by showing the speedup against state-of-art software platform for arbitrary GC problems

# Future Work

- Optimization for better GC performance
- Test on large GC problems
- Garbled Circuits in the Datacenter



# Thank you!

- Reconfigurable and GPU Computing Laboratory
  - http://www.coe.neu.edu/Research/rcl/index.php


- Xin Fang: fang.xi@husky.neu.edu
- Stratis Ioannidis: ioannidis@ece.neu.edu
- Miriam Leeser: mel@coe.neu.edu



Research at Google

ALTERA
now part of Intel

GIDEL