Side-Channel Leakage Evaluation of a RISC V Processor

Chester Rebeiro

Department of Computer Science and Engineering

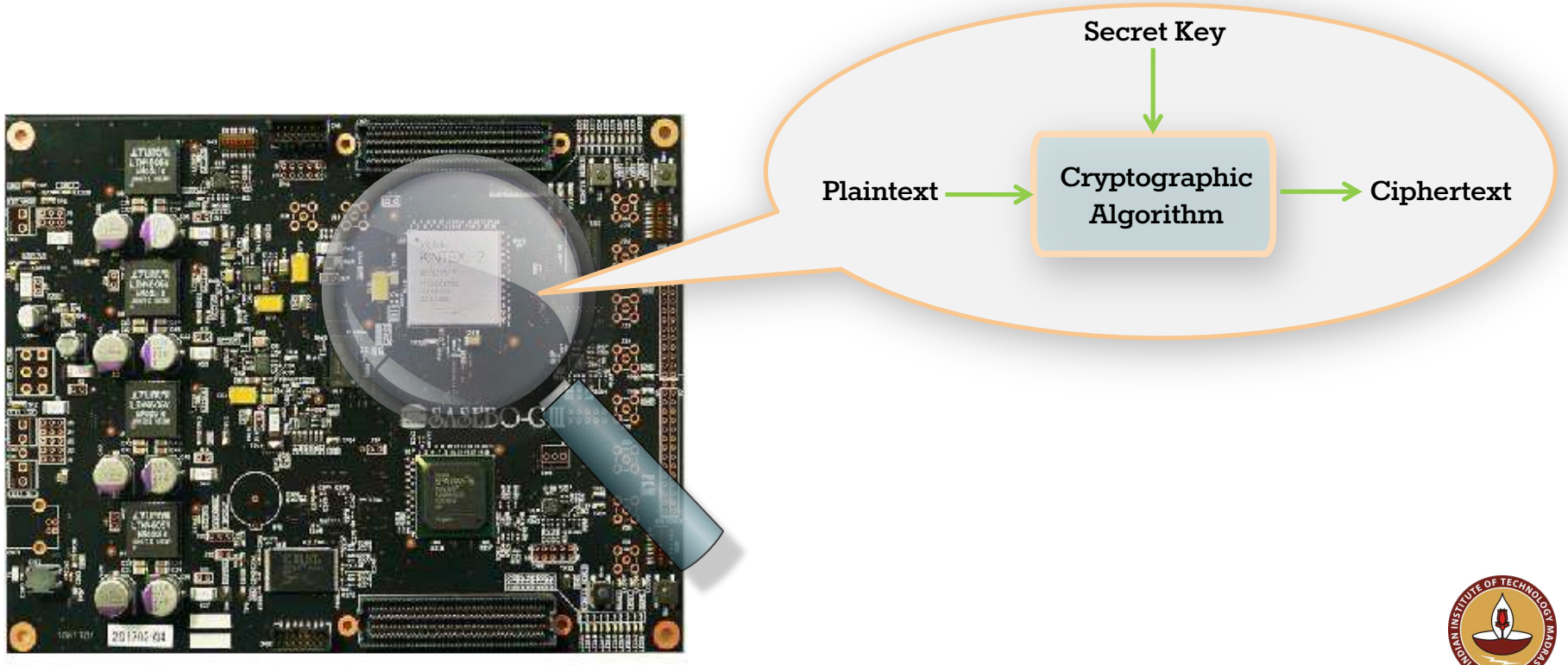
Indian Institute of Technology Madras, India

chester@cse.iitm.ac.in

Side-Channel Analysis

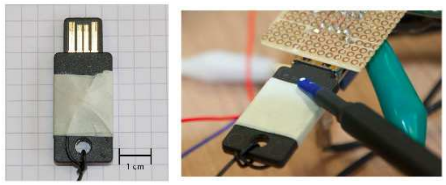
- Attacker can extract secrets from an embedded device through physical parameters

  For example: Device's power consumption, electro-magnetic emanation, execution time, and acoustic information

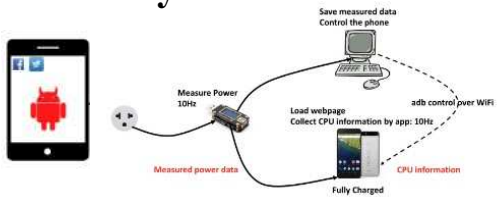


Applications of Side-Channel Analysis

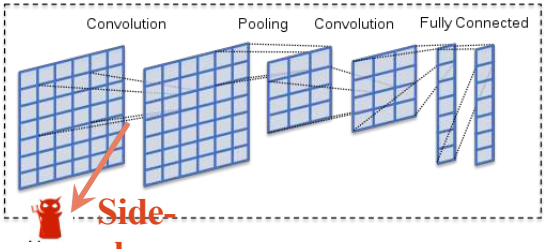
- Firmware reverse engineering
- Recovering passwords
- Mobile app fingerprinting
- Predicting input image used in CNN accelerators

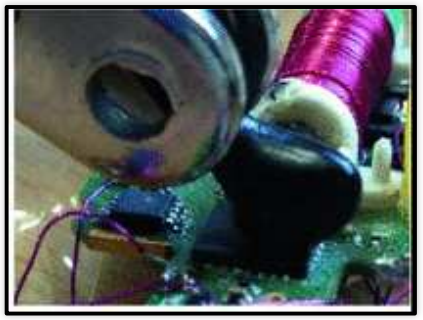


Extracting Secret Keys from Yubikey2 Device



App and Website Fingerprinting from Android Mobiles





EM Probing on PIC Microcontroller

Power Attack Countermeasures

Algorithmic Approach

Implementation Approach

Software Solutions

Hardware Solutions

Limitations of Existing Countermeasures

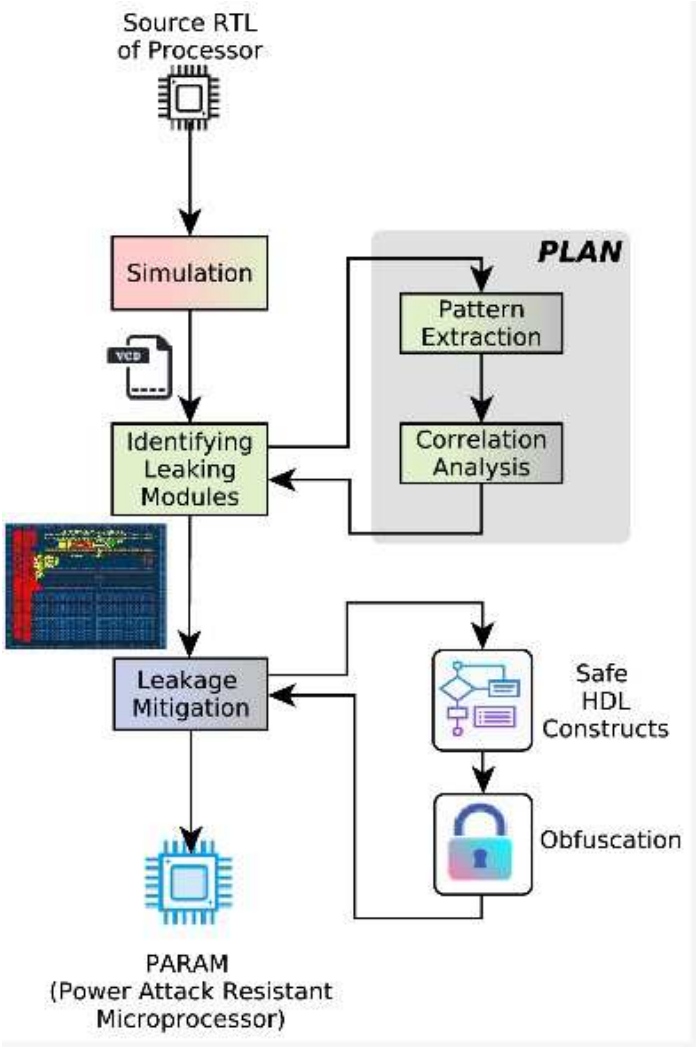
- Algorithm and Implementation solutions:
  - Specific to crypto algorithms
  - Huge overheads up to 3X
- Software solutions do not prevent all leaks
- EDA based approaches cannot provide guarantees due to noise

Computer Architecture Enabled Side-Channel Protection

Empowering computer architects to handle power side-channel leakages

- Countermeasures incorporated early during the design phase
- Minimize overheads
- Side-channel security for all applications executed in a processor

The Big Picture

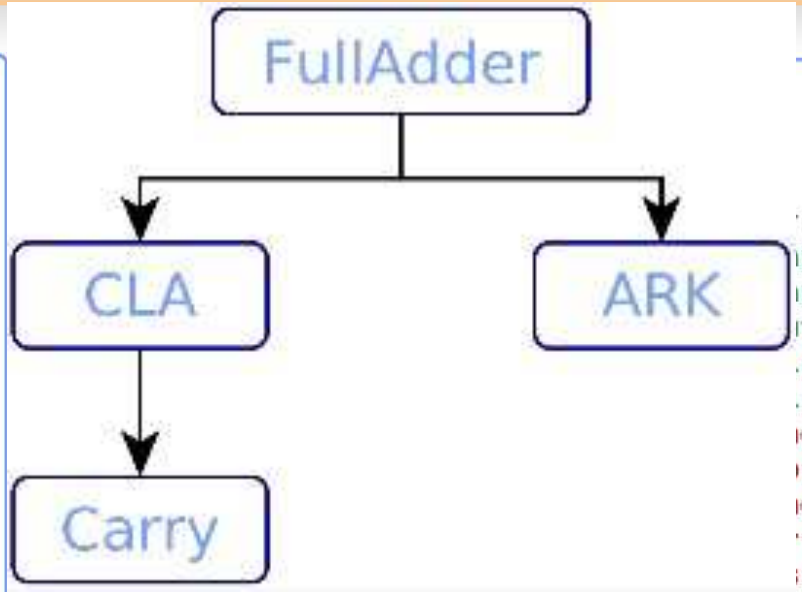


1. Leakage Identification

2. Adding Countermeasures

Leakage Identification: Full Adder

Function of interest: $y=(s+co)\oplus k$




```
module FullAdder;
        // Inputs
        reg [7:0] a;
        reg [7:0] b;
         reg[8:0] k;
        reg c;
         wire [8:0] s,y;
        wire co;
        CLA cla (
                .a(a),
                .b(b),
                .c(c),
                .s(s),
                .co(co)
        );
        ARK ark(
                .s(s),
                .co(co),
                .k(k),
                .y(y)
        );
endmodule
```

```
module ARK(s,co,k,y);
        input [8:0]s,k;
        input co;
        output [8:0]y;
        assign y=(s+co)^k;
endmodule
```

```
ry(a,b,c,s,co);
put a,b;
put c;
tput s,co;
re out4;
re out6;
d (out1,a,b);
r (out2,a,b);
d (out3,out2,c);
 (out4,out3,out1);
sign co = out4;
xor (out5,a,b);
xor (out6,out5,c);
assign s = out6;
endmodule
```

Leakage Identification: Oracle & Side-channel

Oracle Trace

- Derived from function of interest by substituting the actual secret value

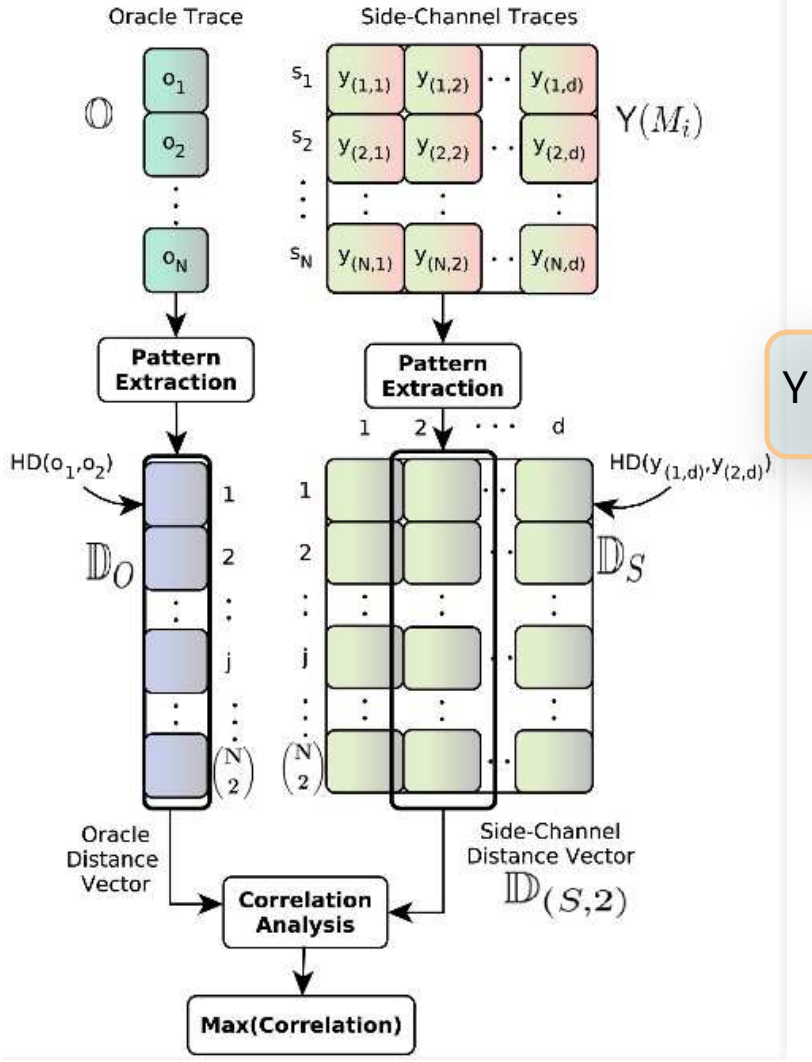
  For example: $(s+co) \oplus k$

- Contains ground truth information

Side-channel Trace

- Derived from simulation data by aggregating value of signals belonging to same module
- Contains approximated power consumption information of each module

PLAN: Power Leakage ANalyzer

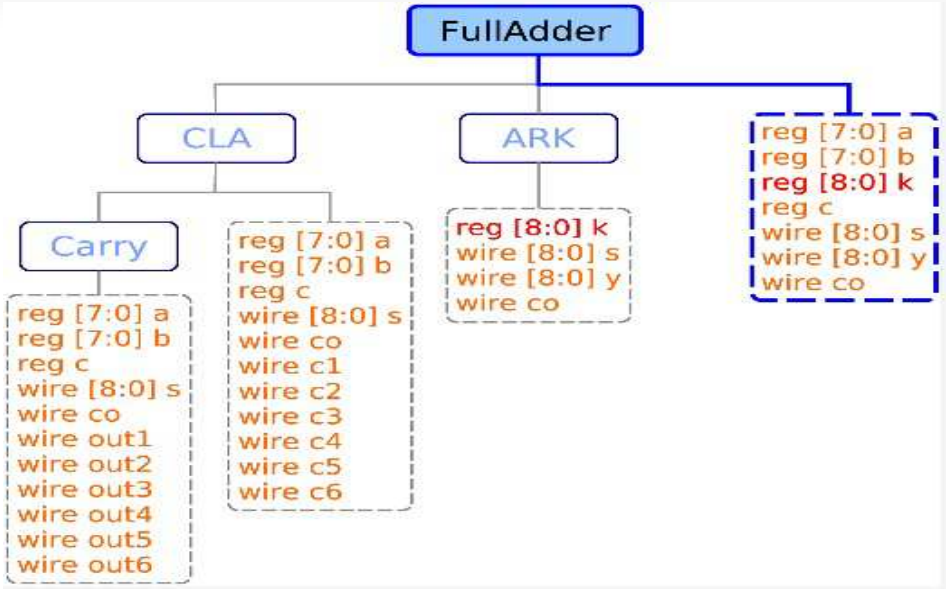


Oracle Trace

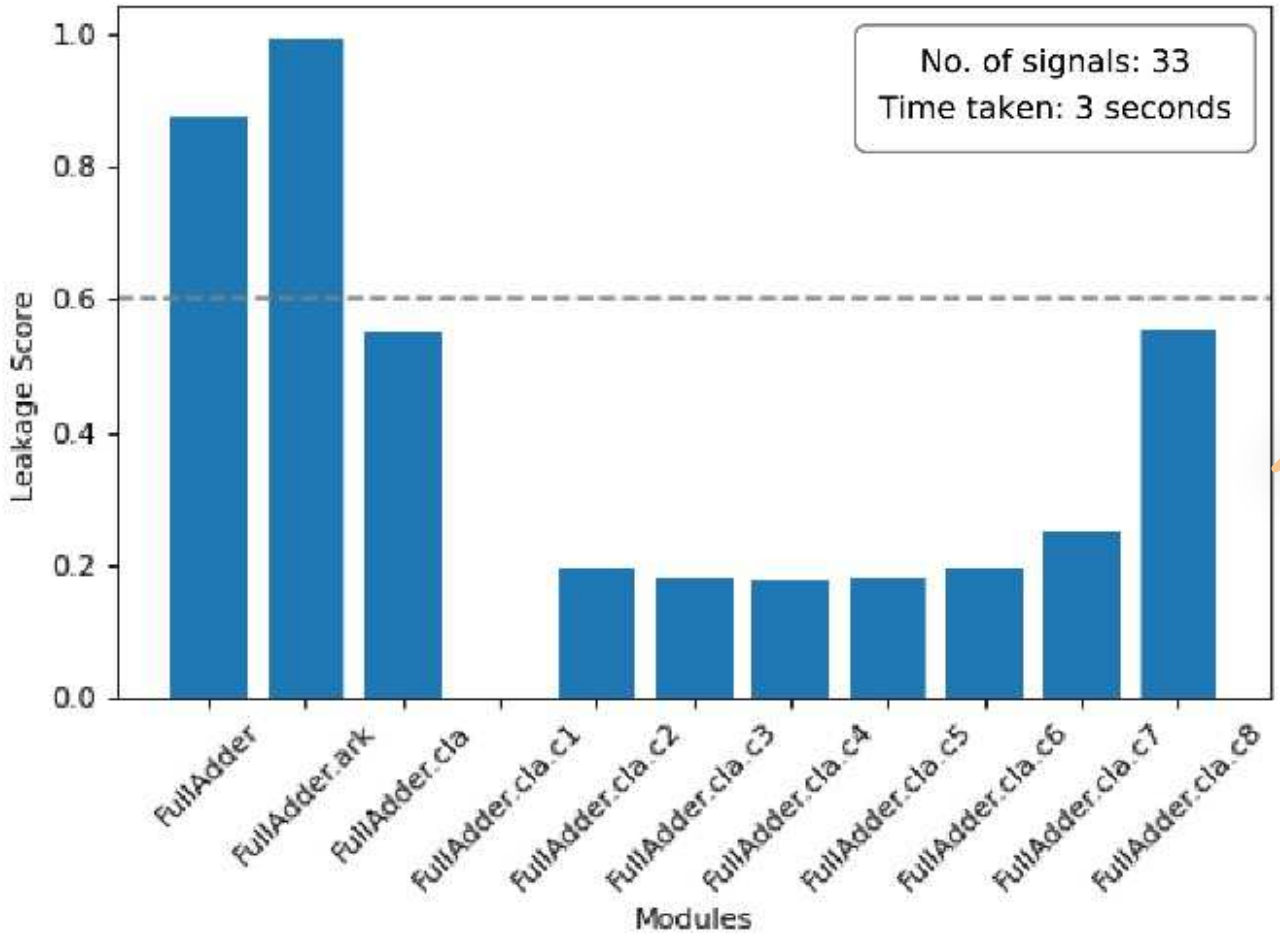
$$O_i = (s+co)_i \oplus k$$

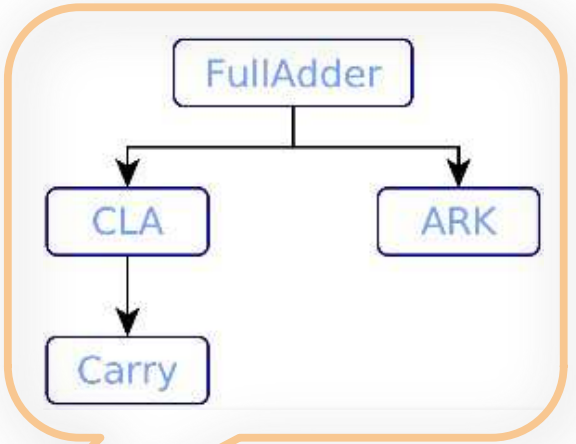
Side-channel Trace

$$Y(FullAdder)_{(i,j)} = (a||b||c||co||k||s||y)_{(i,j)}$$



Information Leakage: Full Adder



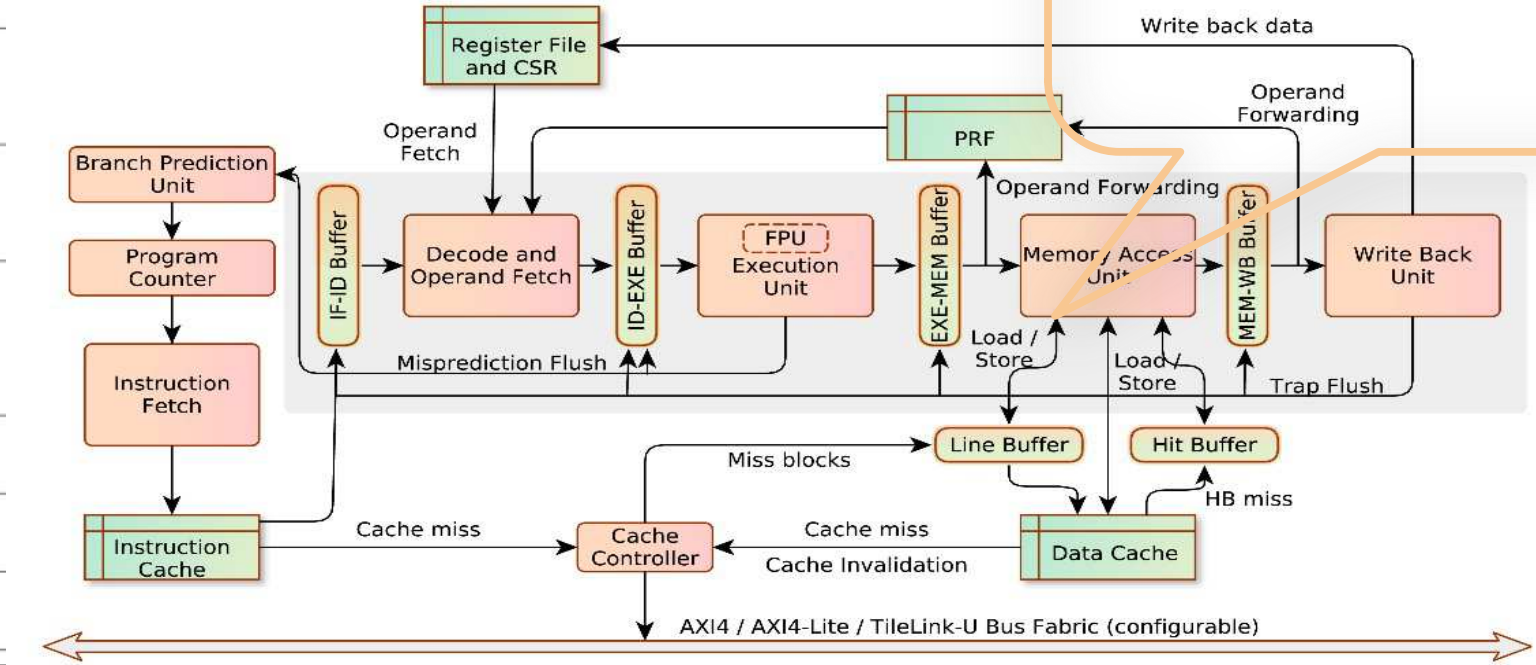


Leakage in a RISC V Processor: Shakti C[1]

Specifications

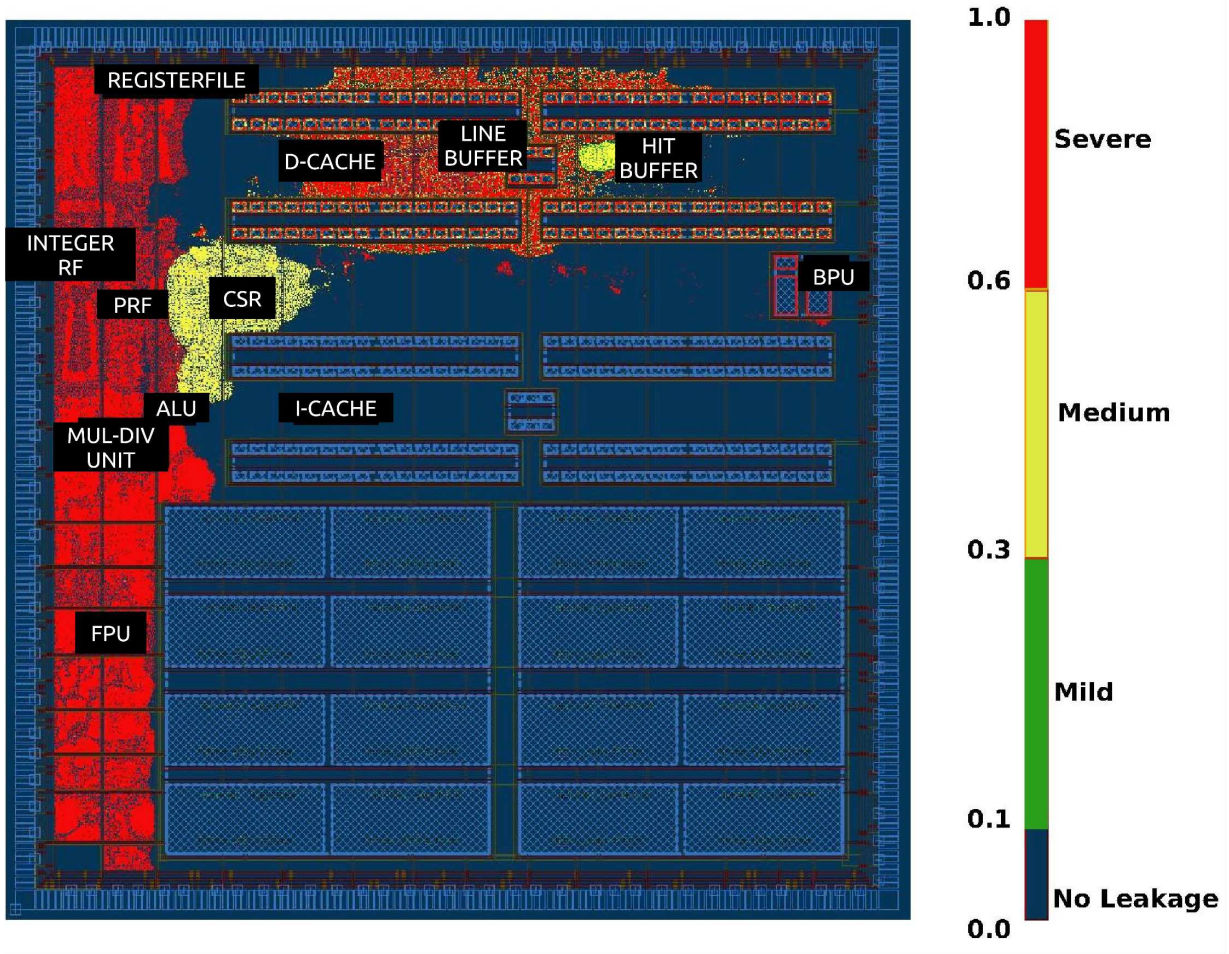
- RISC V, 5 stage processor
- Verilog source code consists of
  ls

	Module	Leak Count	SVF	#Leaking Signals
	Data Cache	165	0.99	

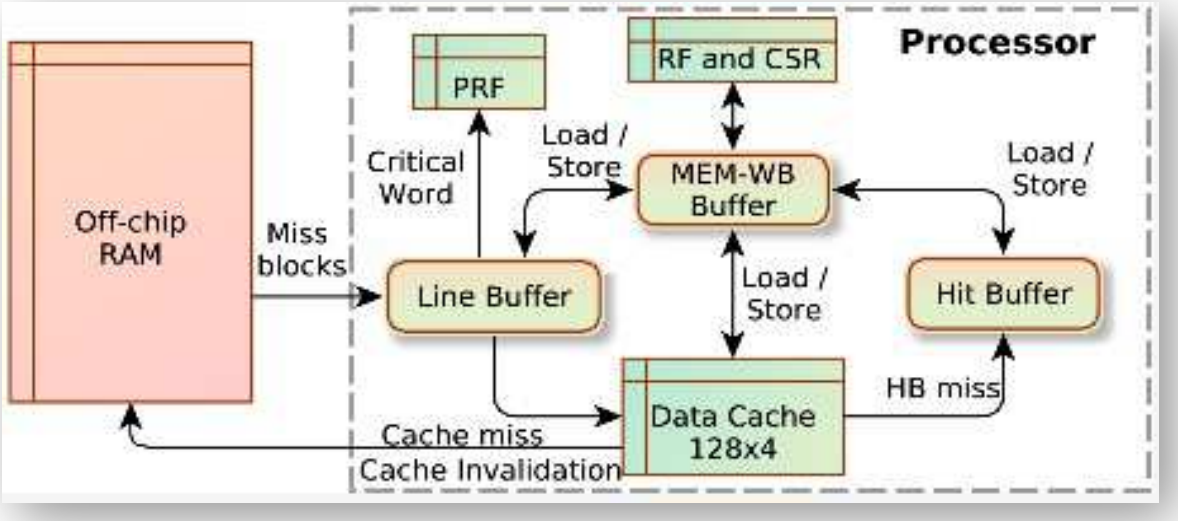


[1] SHAKTI processors: An open-source hardware initiative, VLSID 2016.

Information Leakage: Shakti-C

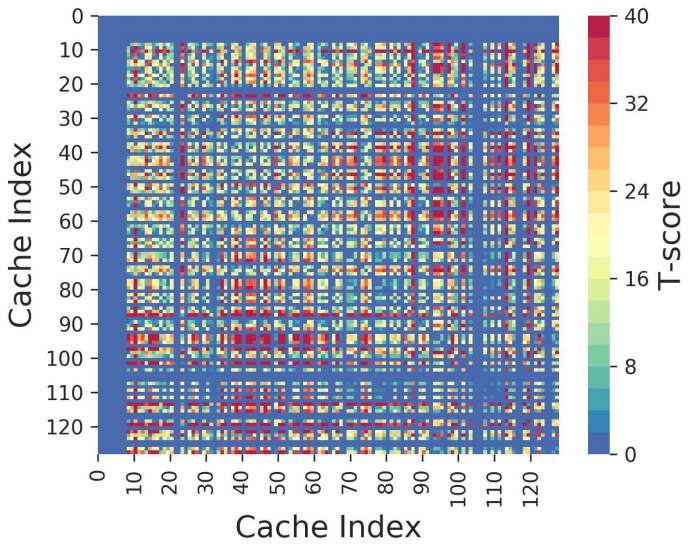


Analysis of Memory Units

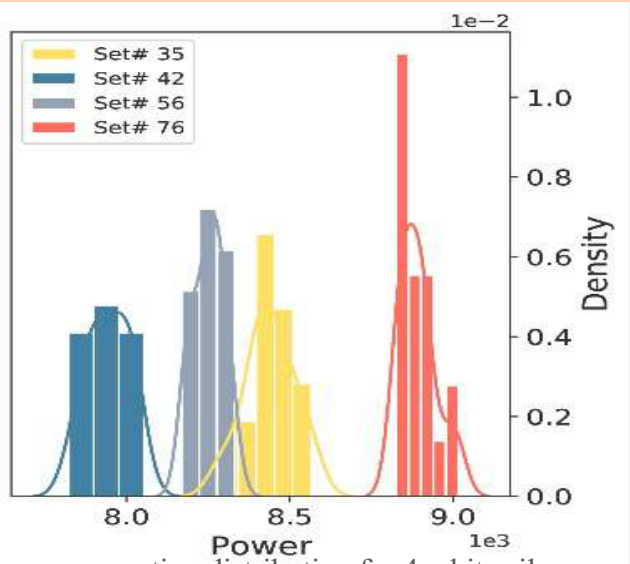


Data paths of Shakti-C interconnects the storage structures

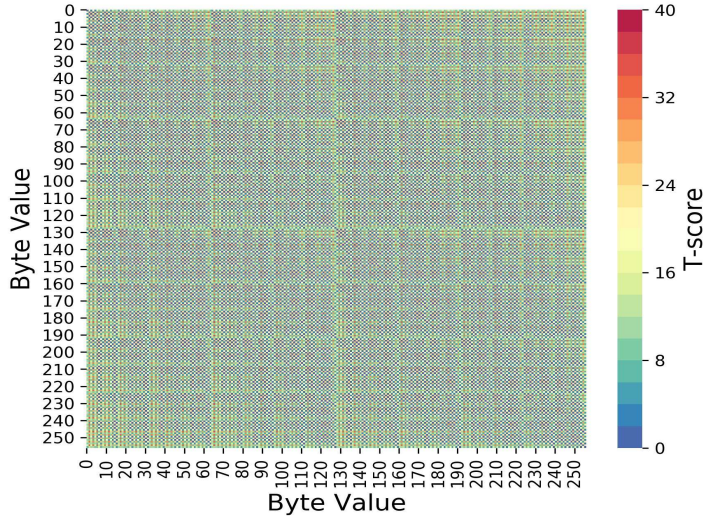
Data Cache Leakages



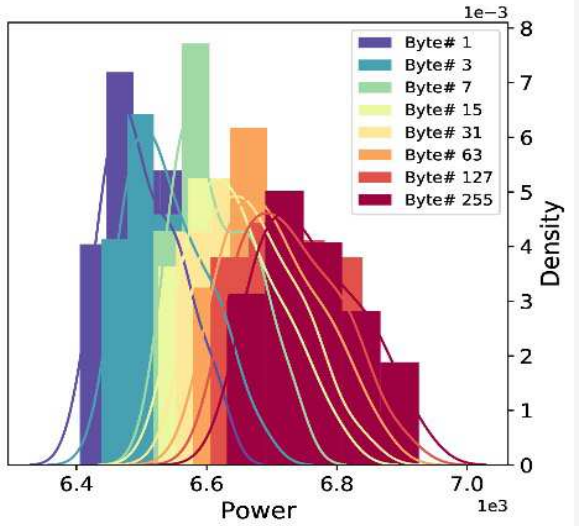
Pairwise differential power consumption of cache sets



Power consumption distribution for 4 arbitrarily chosen cache sets

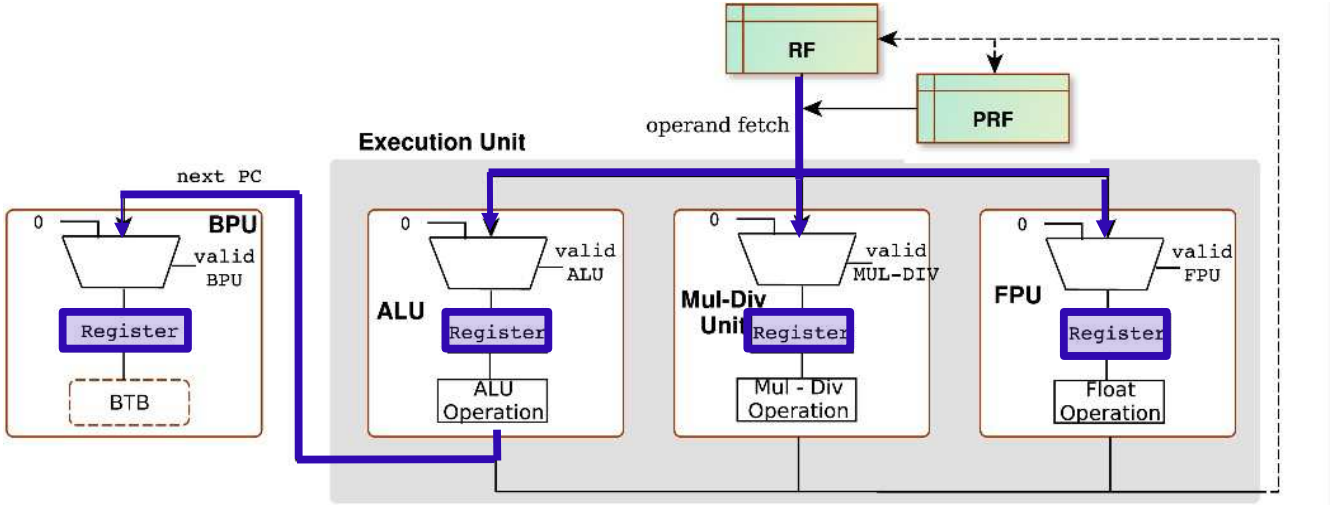


Pairwise differential power consumption for data stored in a cache set

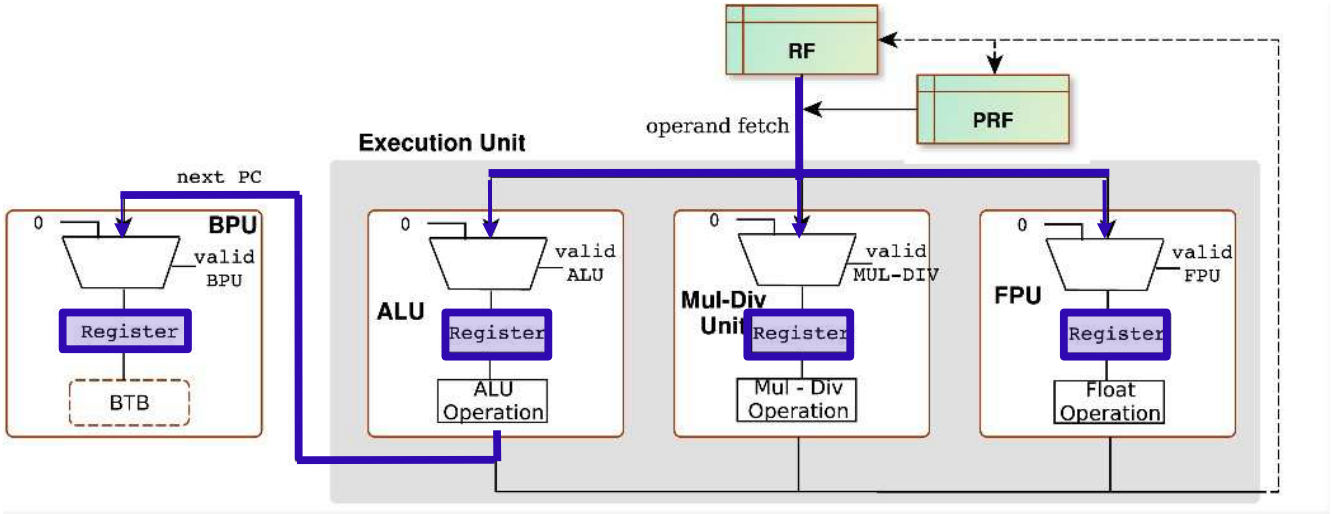


Power consumption distribution for arbitrarily chosen data stored in a cache set

Analysis of Leakage in Functional Units



Expected design of Execution Unit in reference platform Shakti-C



Design after Bluespec compilation

Fixing the Leakages

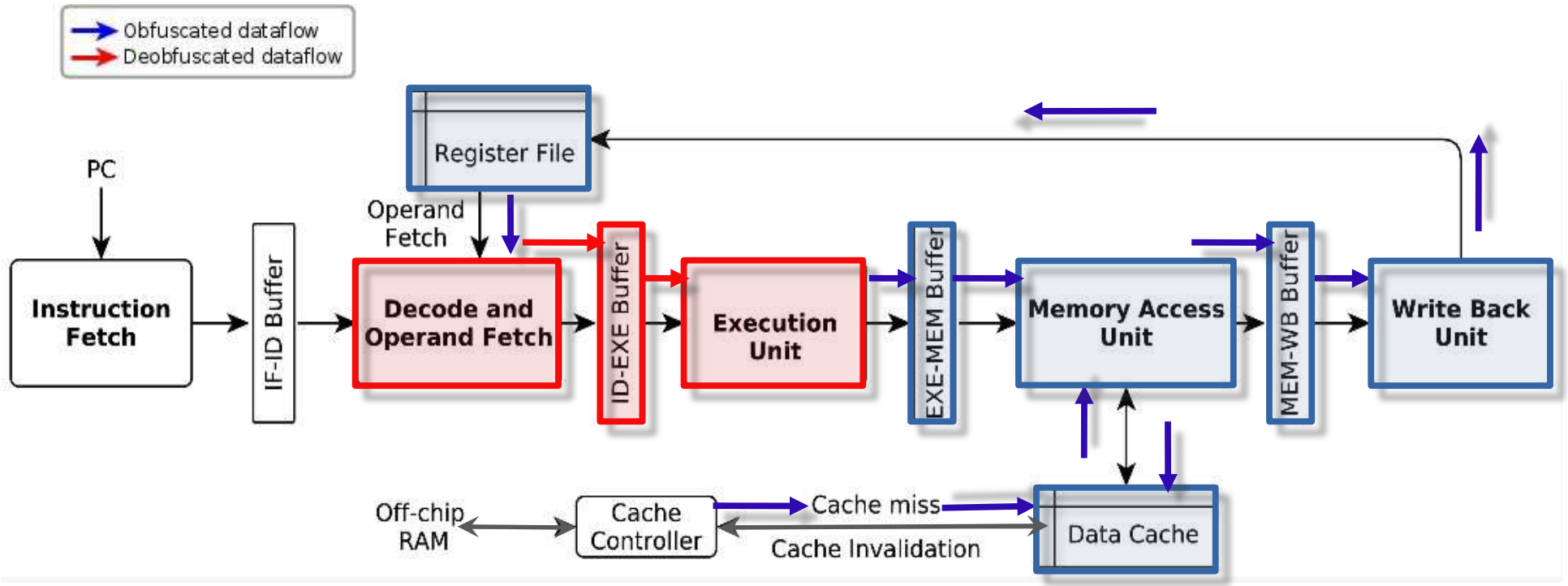
1. Modules holding data correlated with the secret

Obfuscate data to break the correlation between data and power consumption

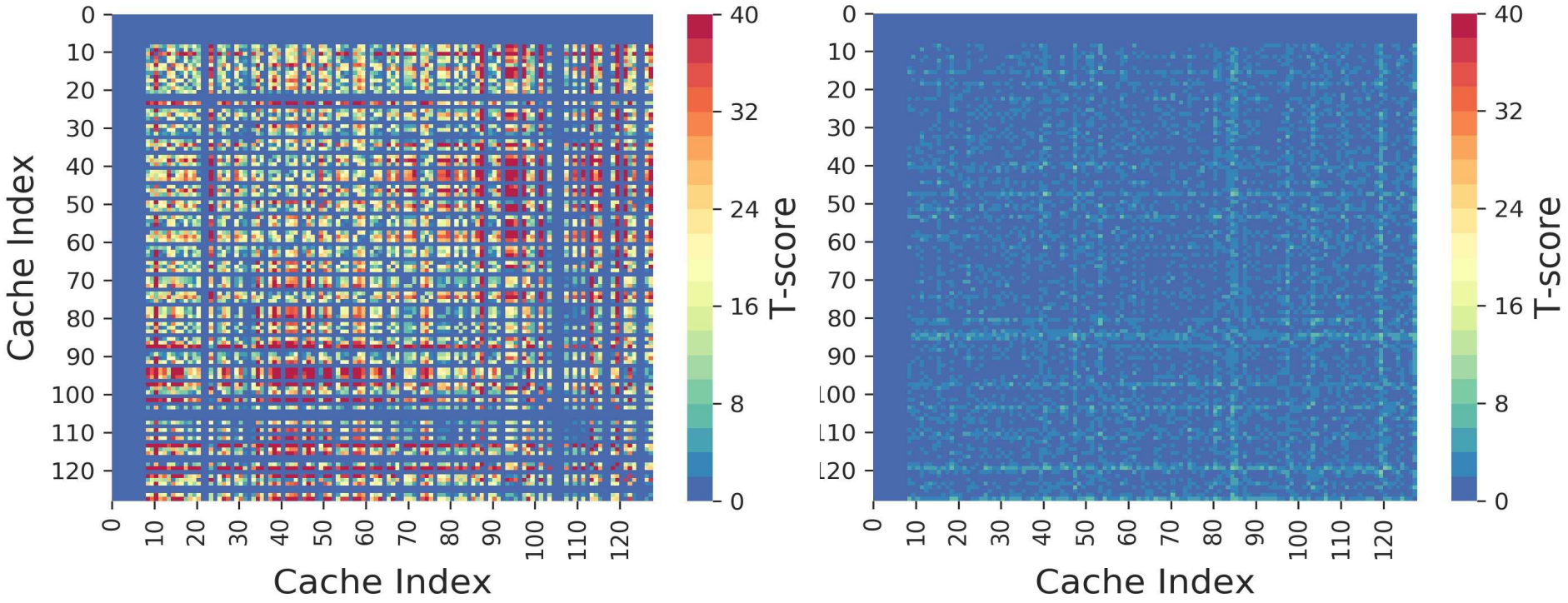
2. Security agnostic EDA translations

Use safe HDL constructs

Fixing Correlation with Obfuscation



Leakage Reduction in Data Cache



Fixing Leakage due to Security Agnostic EDA Translation

```
rule rl_addition;
  if(a1 > 'h20)
    rg_out <= tagged Valid (a1+a2);
  else
    rg_out <= tagged Invalid;
Endrule
/* A combinational block then reads
rg_out and checks valid bit before
using it */
```

BSV to Verilog Translation by Bluespec Compiler

```
wire rg_out$D_IN;
reg rg_out;

assign rg_out$D_IN =
{a1>'h20, a1+a2};

always@(posedge clk) begin
  rg_out <= rg_out$D_IN;
end

assign out = rg_out[31:0];
assign RDY_out = rg_out[32];
```

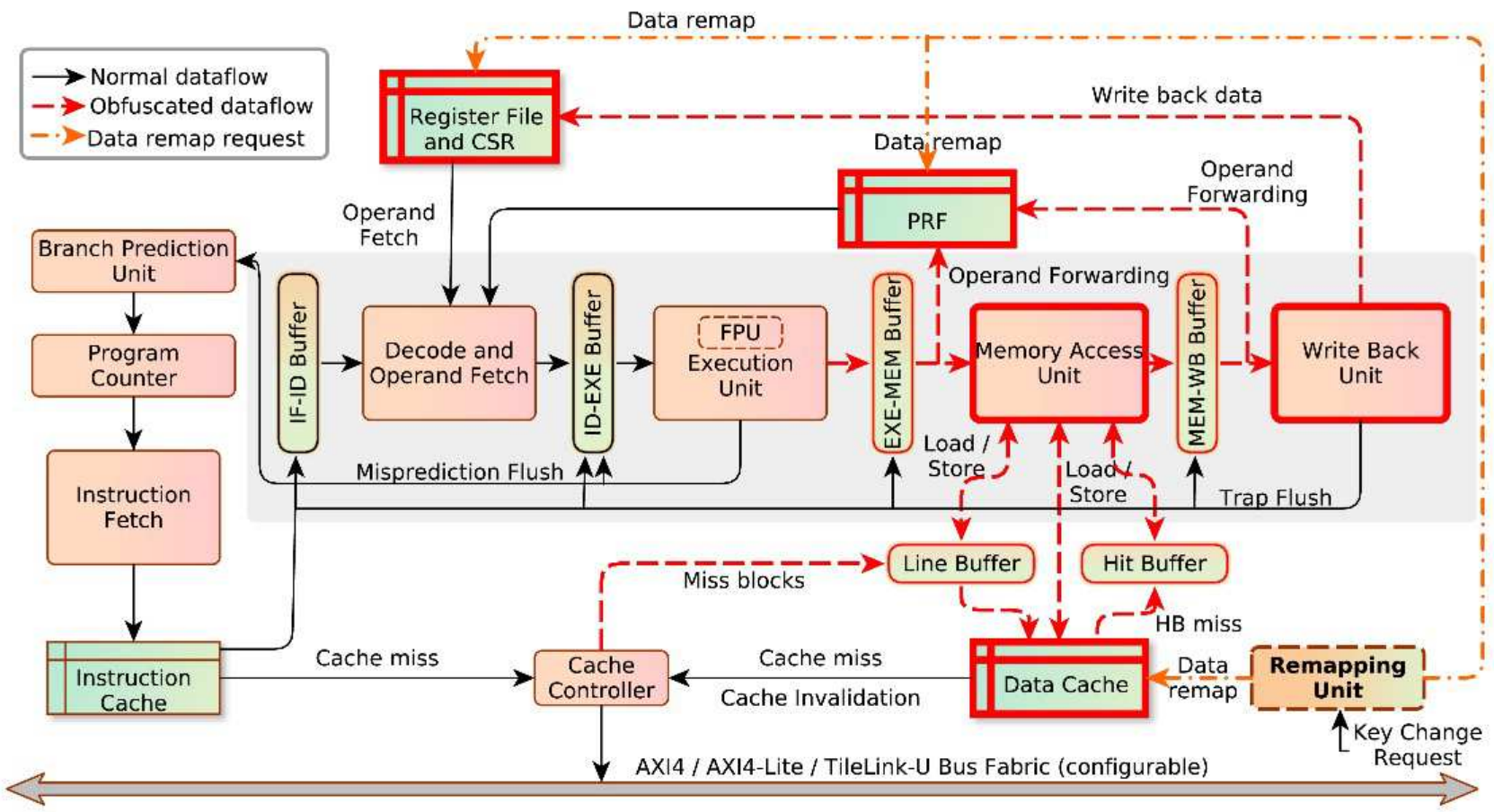
```
rule rl_addition;
  if(a1 > 'h20)
    rg_out <= {1'b1, (a1+a2)};
  else
    rg_out <= {1'b0, 32'd-1};
Endrule
/* A combinational block then reads
rg_out and checks valid bit before
using it */
```

BSV to Verilog Translation by Bluespec Compiler

```
wire rg_out$D_IN;
reg rg_out;

assign rg_out$D_IN = a1 > 'h20 ?
{1'b1, a1+a2} : 33'h0FFFFFFF;

always@(posedge clk) begin
  rg_out <= rg_out$D_IN;
end

assign out = rg_out[31:0];
assign RDY_out = rg_out[32];
```

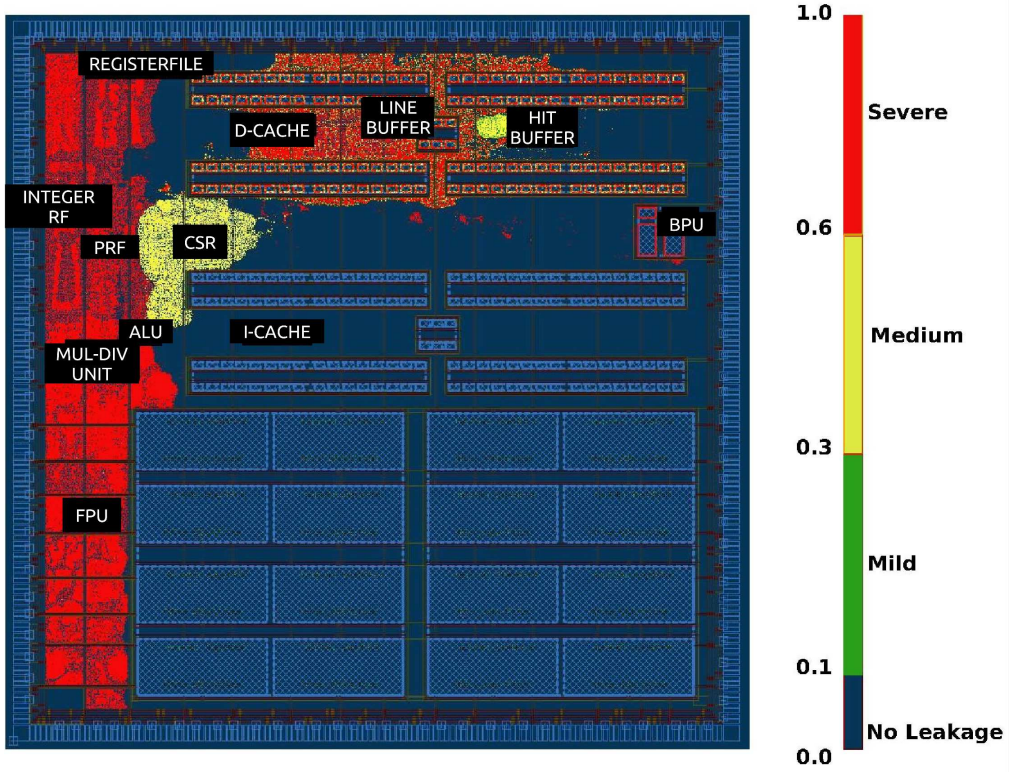
An illustration of leakages due to EDA translations before and after fix

PARAM: Power Attack ResistAnt Microprocessor

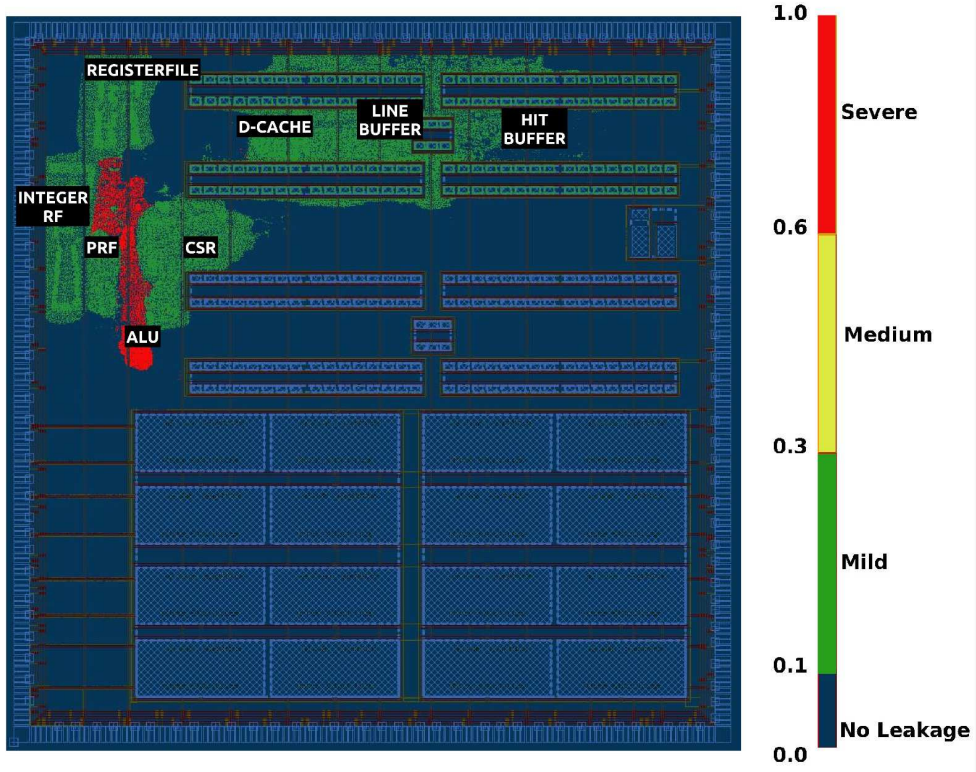


Information Leakage Plots

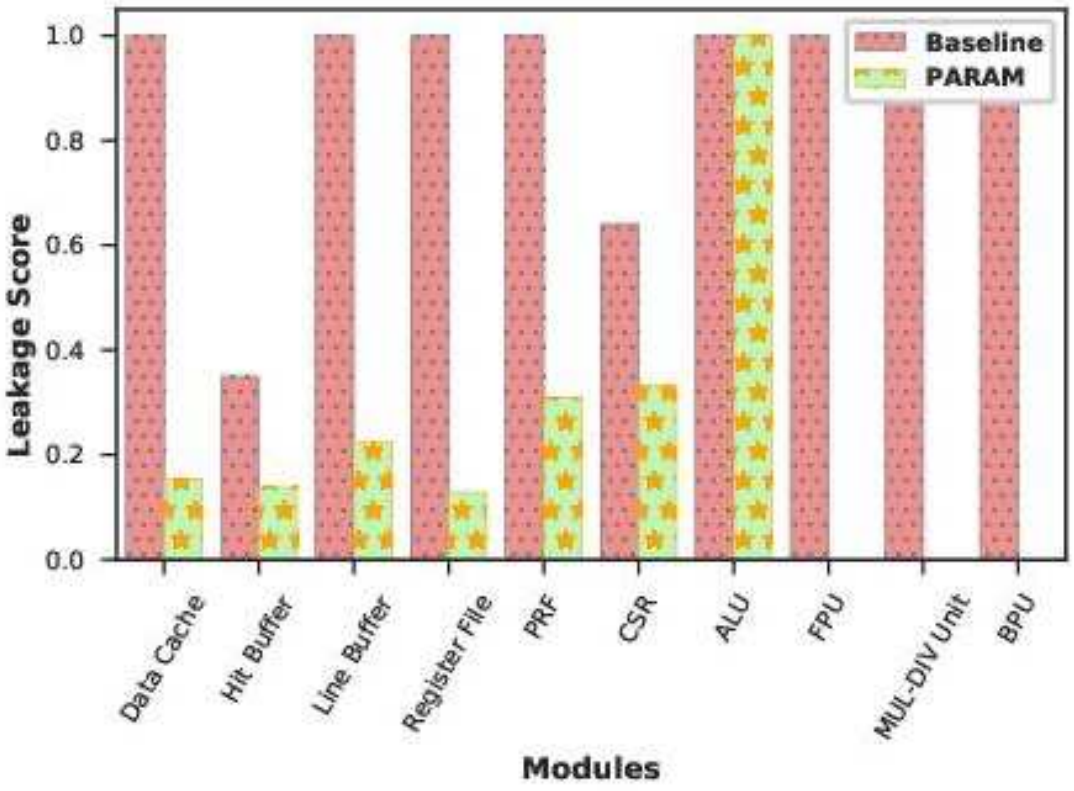
Shakti-C



PARAM



Results

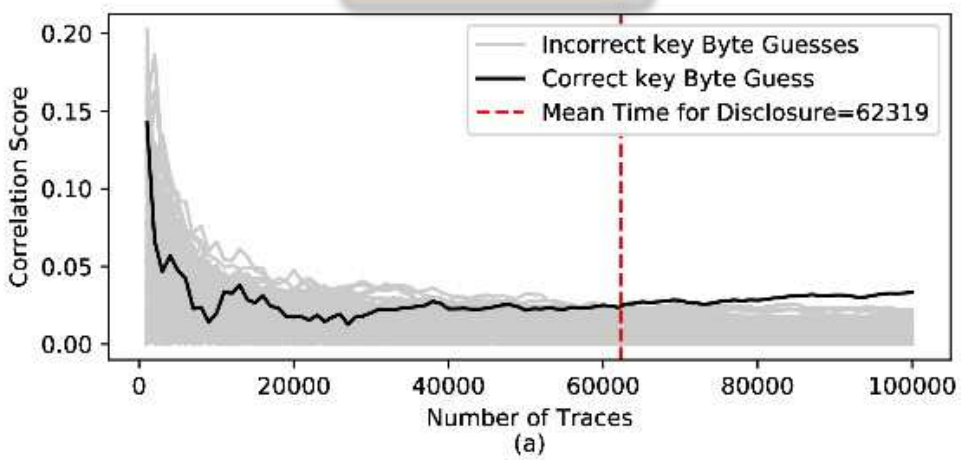

Baseline
PARAM
Leakage Score
1.0
0.8
0.6
0.4
0.2
0.0
Data Cache
Hit Buffer
Line Buffer
Register File
PRF
CSR
ALU
FPU
MUL-DIV Unit
BPU
Modules

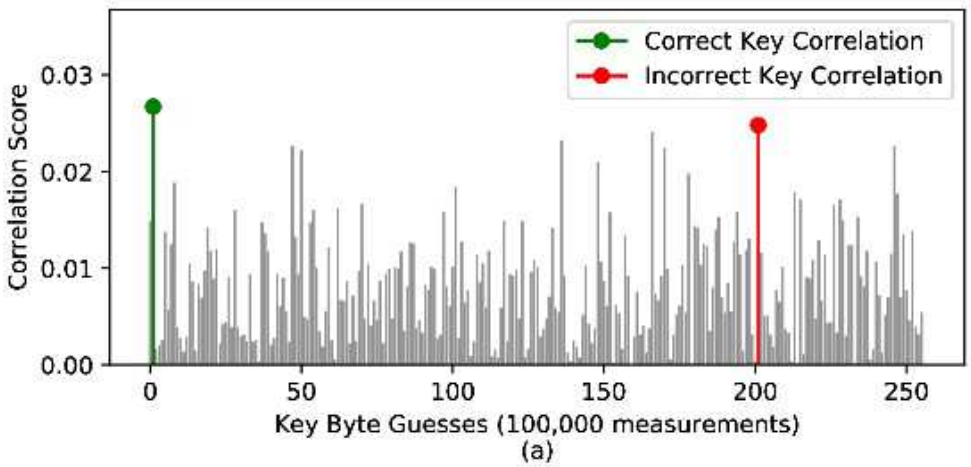
Experimental Setup: Differential Power Analysis

- Processor is instantiated on the SASEBO-GIII FPGA board which has Kintex-7 (C7K160T-1FBG676C) FPGA
- Collect power traces while executing benchmark code on both baseline and PARAM processors
- Compute Mean Time for key Disclosure (MTD) to measure resistance against first-order DPA

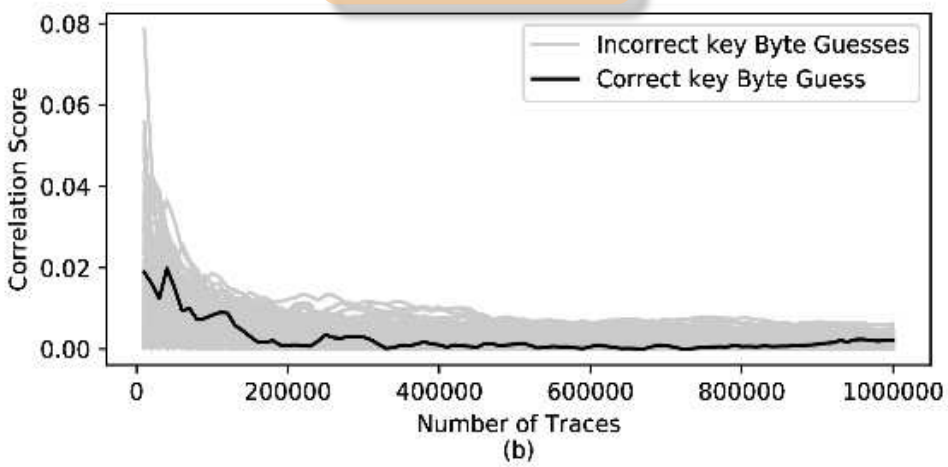
DPA Results

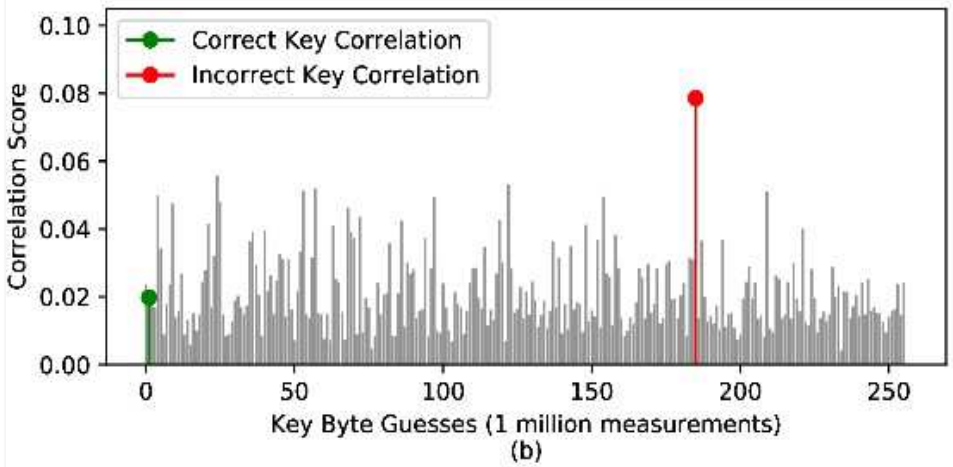
Shakti-C





PARAM





DPA results of reference architecture before and after leakage mitigation

Conclusion and Future works

- First general purpose processor with built-in security against power attacks
- Low area and performance overheads
- Address ALU and control path leaks

Questions ?

Email: chester@cse.iitm.ac.in