Bending microarchitectural weird machines towards practicality

Ping-Lun Wang, Riccardo Paccagnella, Riad S. Wahby,
and Fraser Brown, *Carnegie Mellon University*

https://www.usenix.org/conference/usenixsecurity24/presentation/wang-ping-lun

How to identify these kinds of weird machines

FaultDetective

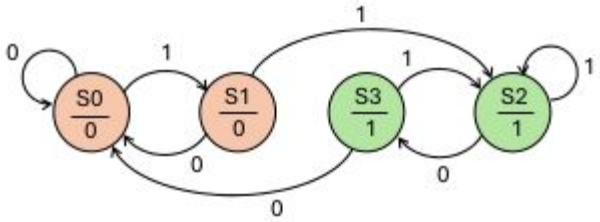
Explainable to a Fault, from the Design Layout to the Software

Zhenyuan Liu[1], Dillibabu Shanmugam[1] and Patrick Schaumont[1]

Worcester Polytechnic Institute, Worcester, MA 01609, USA,
{zliu12,dshanmugam,pschaumont}@wpi.edu

Automate anomaly detection

FSM_RED




```
# TARGET FIRMWARE
export TARGET = fsm_red

# GLITCH WIDTH
export GLITCH_WIDTH_HIGH = 1.060
export GLITCH_WIDTH_LOW = 1

# TOTAL CYCLE FROM TRIGGER UP TO DOWN, FROM 0
export TOTAL_CYCLE = 27

# AFTER HOW MANY CYCLE TRIGGER STARTS, FROM 0
export TRIGGER_OFFSET = 7

# HOW MANY CLOCKS TO RUN IN TESTBENCH (DO NOT TOUCH)
export CLOCK_TO_RUN = 120
```

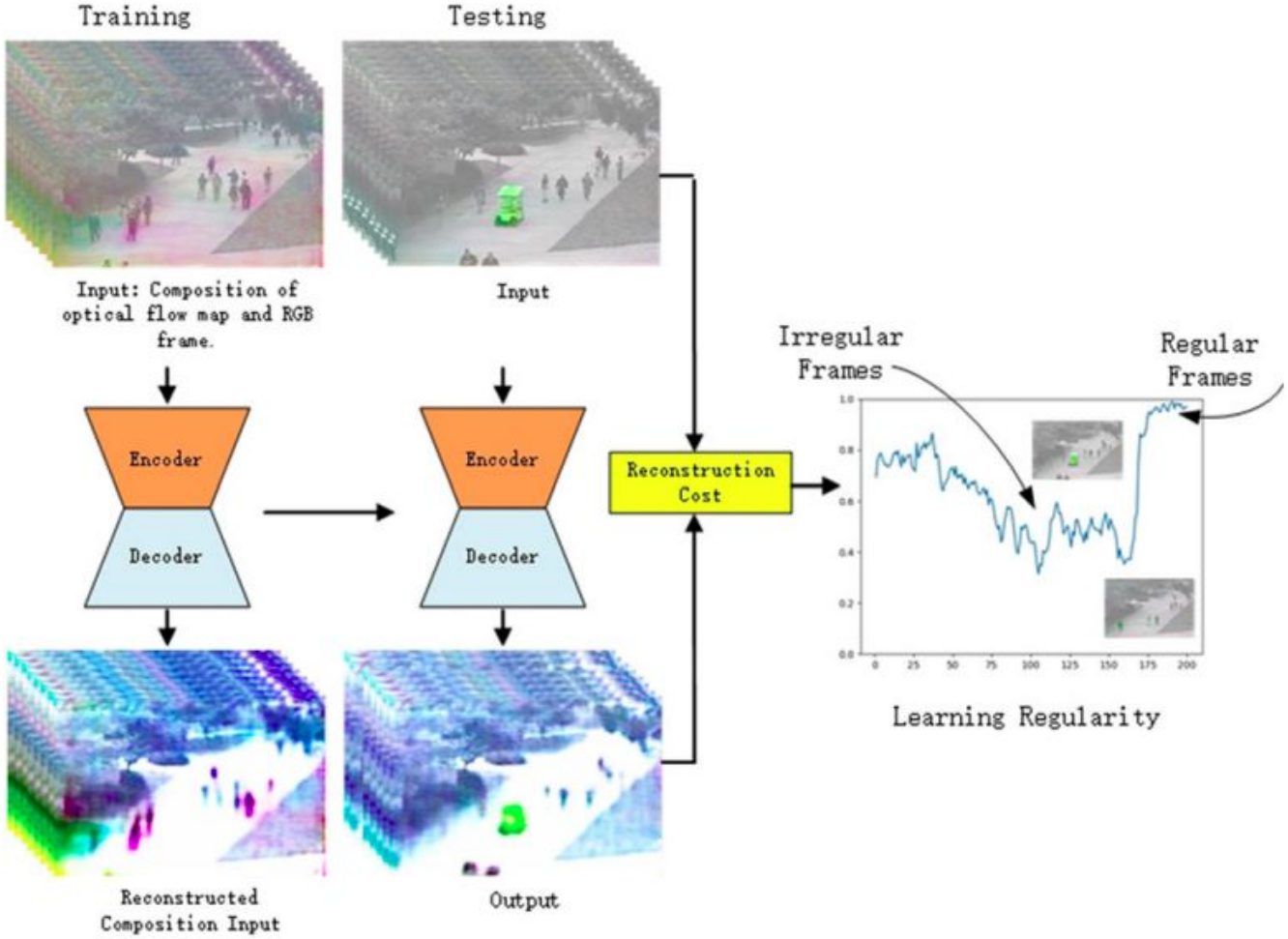
Glitchsweep fault injection trigger high for 27 clock cycles

660 processor bits to 31 features

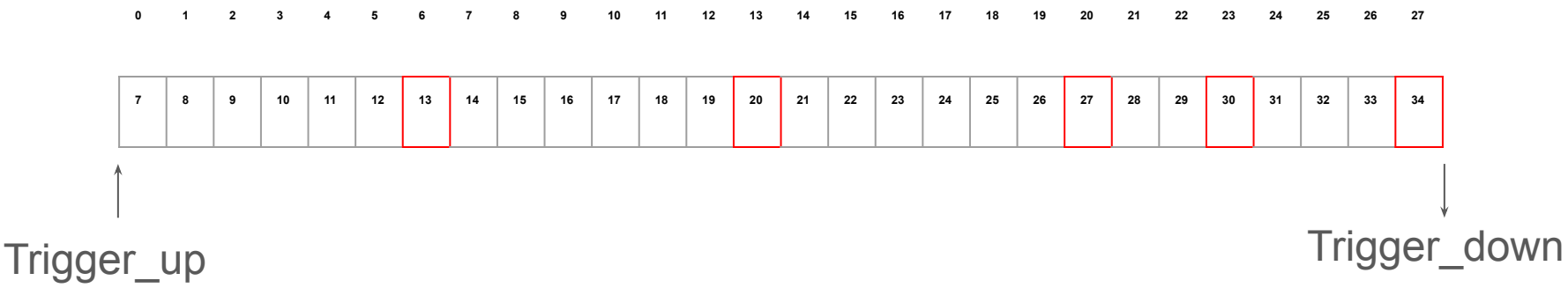
Feature extracted

1. msp430Inst_0
2. msp430Inst_0.openmsp430_0
3. Openmsp430_0.watchdog_0
4. Openmsp430_0.sfr_0
5. Openmsp430_0.multiplier_0
6. Openmsp430_0.mem_backbone_0
7. Openmsp430_0.frontend_0
8. Openmsp430_0.execution_unit_0
9. Openmsp430_0.dbg_0
10. Openmsp430_0.clock_module_0
11. Openmsp430_0.archi
12. Openmsp430_0.watchdog_0.wdtctl
13. Openmsp430_0.sfr_0.wdtie_reg
14. Openmsp430_0.sfr_0.nmiifg_reg
15. Openmsp430_0.sfr_0.nmie_reg
16. Openmsp430_0.sfr_0.nmi_dly_reg
17. Openmsp430_0.multiplier_0.sumext_s
18. Openmsp430_0.multiplier_0.reslo
19. Openmsp430_0.multiplier_0.reshi
20. Openmsp430_0.multiplier_0.op2
21. Openmsp430_0.multiplier_0.op1
22. Openmsp430_0.frontend_0.pc
23. Openmsp430_0.execution_unit_0.register_file_0
24. Openmsp430_0.dbg_0.mem_data
25. Openmsp430_0.dbg_0.mem_ctl
26. Openmsp430_0.dbg_0.mem_cnt
27. Openmsp430_0.dbg_0.mem_addr
28. Openmsp430_0.dbg_0.cpu_stat
29. Openmsp430_0.dbg_0.cpu_ctl
30. Openmsp430_0.clock_module_0.bcsctl2
31. openmsp430_0.clock_module_0.bcsctl1

Anomaly detection



FSM_RED



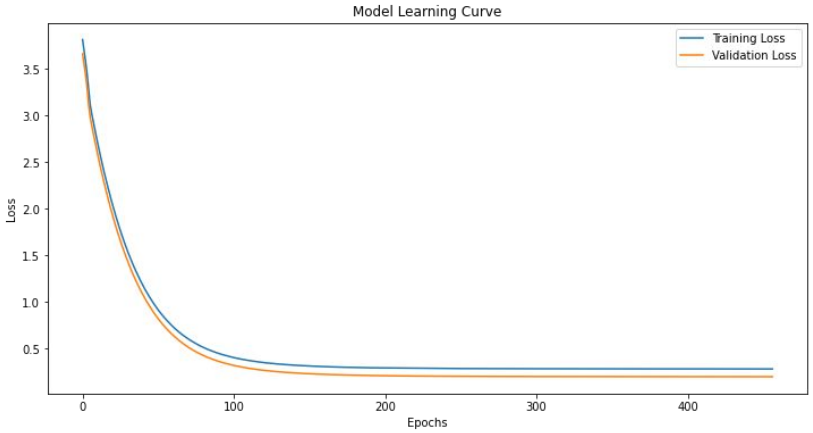
The trigger is set high for 28 clock cycles (28cc). Faults were injected across all clock cycles, but the injections were effective only at clock cycles 13, 20, 27, 30, and 34.

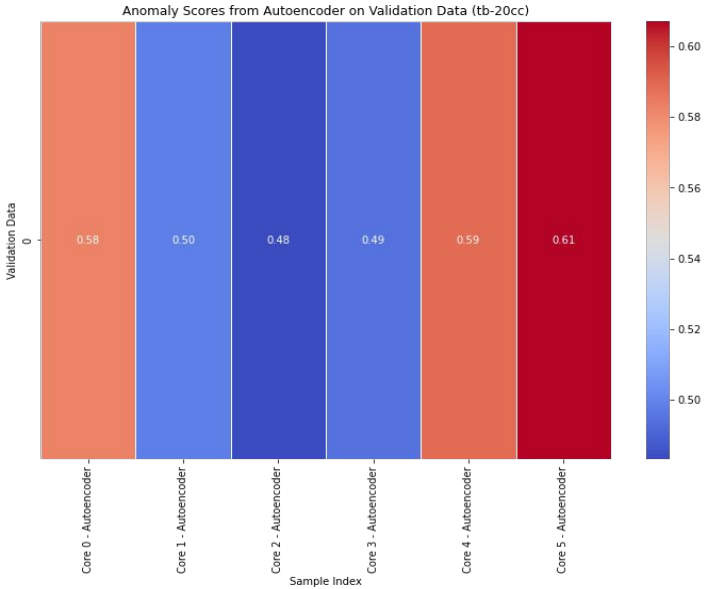
Anomaly detection

28 cc * 31 features

Training dataset
138 * (28, 31)

Testing dataset
6 * (28, 31)



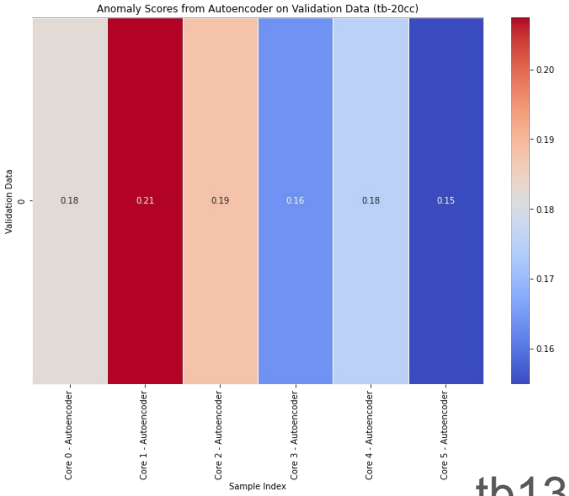


Faulty test cases

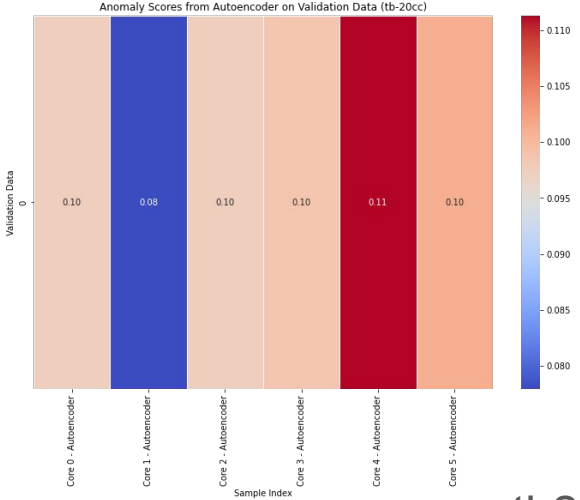
tb13,tb20,tb27,tb30,tb34

core/tb	13	20	27	30	34
0	x	x			
1		x	x		
2					x
3					x
4	x	x		x	x
5	x				

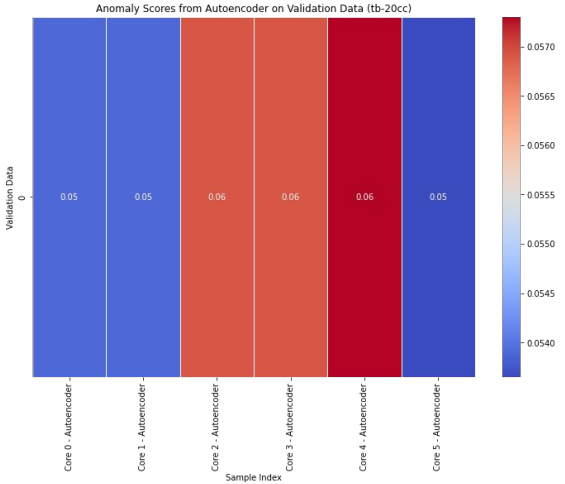
tb27



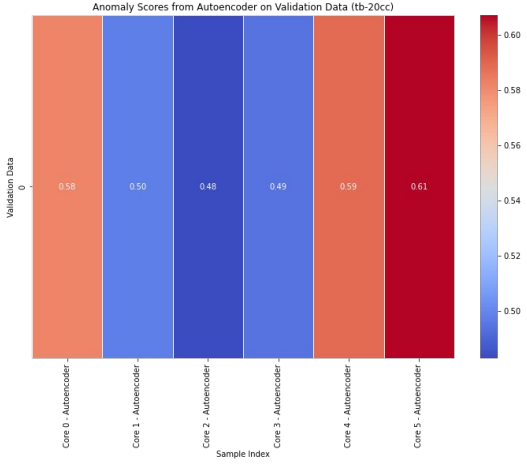
tb30



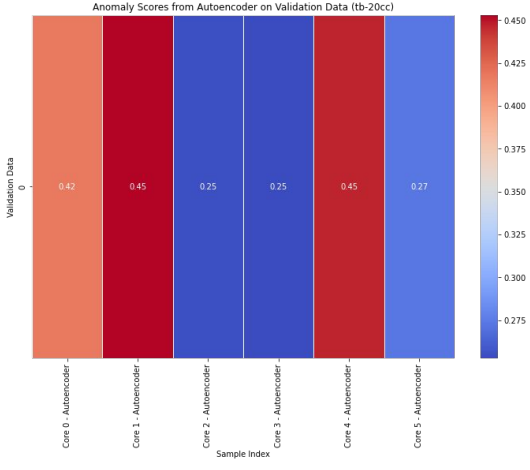
tb34



tb13

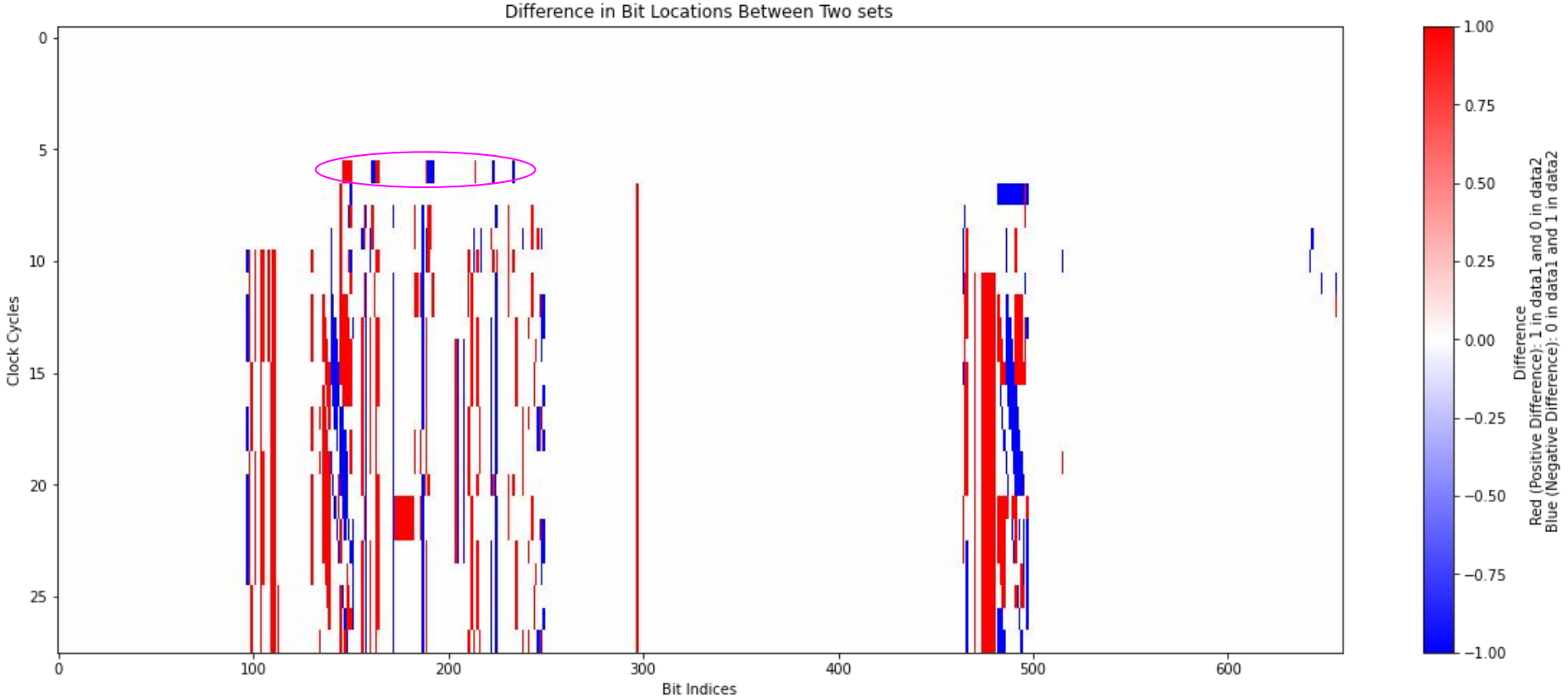


tb20



Tb13 : core_4

Compare faulty(tb13_core4) with non-faulty core(tb-correct_core1)



The first difference occurs at clock cycle: 6

660 Processor bits

Use regular expression to categorize these signals and extract bits

```
1 bit_diff_signal_names
```

```
146              frontend_0.pc_reg_5_.Q;
147              frontend_0.pc_reg_4_.Q;
148              frontend_0.pc_reg_3_.Q;
149              frontend_0.pc_reg_2_.Q;
150              frontend_0.pc_reg_1_.Q;
161    frontend_0.inst_src_bin_reg_3_.Q;
162    frontend_0.inst_src_bin_reg_2_.Q;
163    frontend_0.inst_src_bin_reg_1_.Q;
164    frontend_0.inst_src_bin_reg_0_.Q;
189           frontend_0.inst_mov_reg.Q;
190    frontend_0.inst_jmp_bin_reg_2_.Q;
191    frontend_0.inst_jmp_bin_reg_1_.Q;
192    frontend_0.inst_jmp_bin_reg_0_.Q;
214            frontend_0.inst_bw_reg.Q;
223       frontend_0.inst_alu_reg_11_.Q;
233        frontend_0.inst_alu_reg_1_.Q;
234        frontend_0.inst_alu_reg_0_.Q;
```

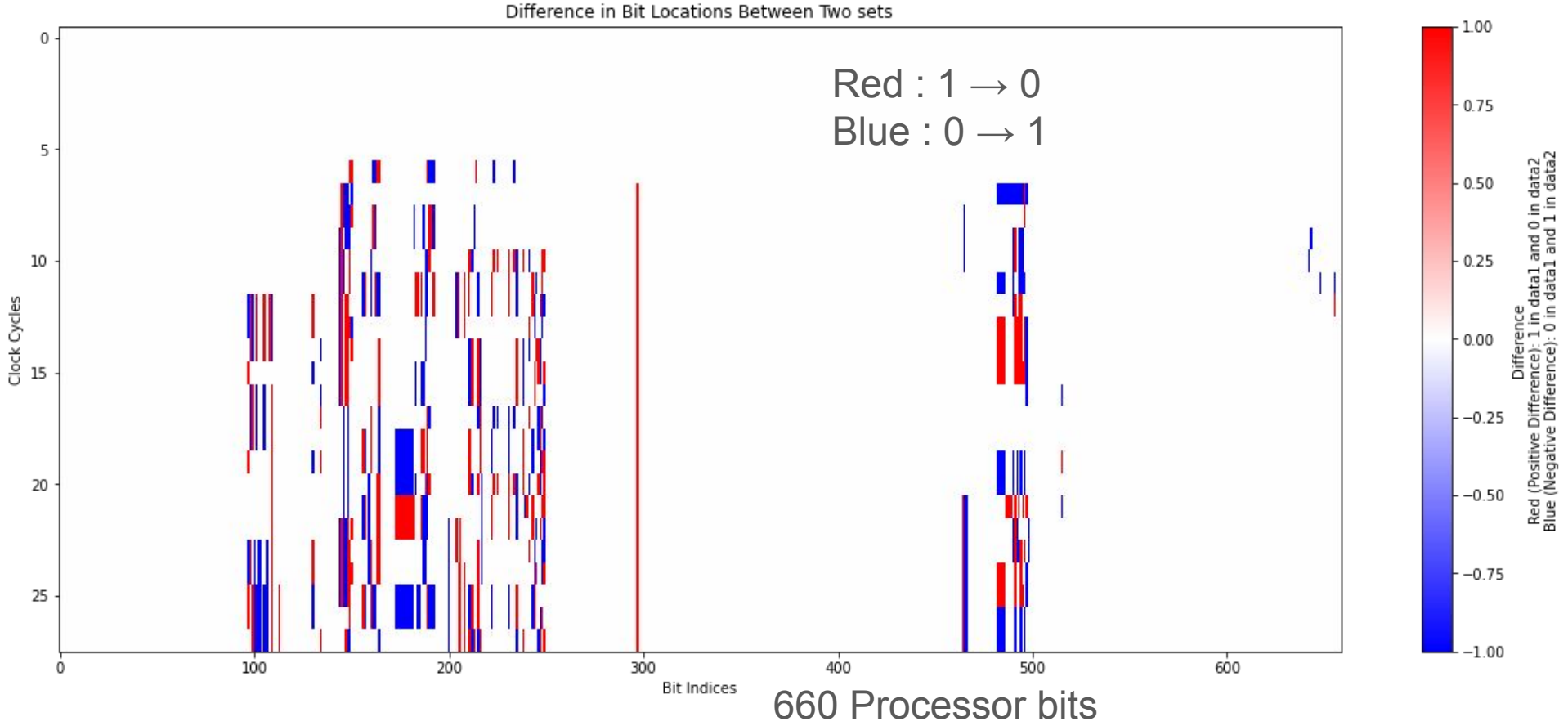
```
Categorized Signals:
frontend_0.pc_reg = ['frontend_0.pc_reg_5_.Q;', 'frontend_0.pc_reg_4_.Q;', 'frontend_0.pc_reg_3_.Q;',
'frontend_0.pc_reg_2_.Q;', 'frontend_0.pc_reg_1_.Q;']
frontend_0.inst_src_bin_reg = ['frontend_0.inst_src_bin_reg_3_.Q;', 'frontend_0.inst_src_bin_reg_2_.
Q;', 'frontend_0.inst_src_bin_reg_1_.Q;', 'frontend_0.inst_src_bin_reg_0_.Q;']
frontend_0.inst_mov = ['frontend_0.inst_mov_reg.Q;']
frontend_0.inst_jmp_bin_reg = ['frontend_0.inst_jmp_bin_reg_2_.Q;', 'frontend_0.inst_jmp_bin_reg_1_.
Q;', 'frontend_0.inst_jmp_bin_reg_0_.Q;']
frontend_0.inst_bw_reg = ['frontend_0.inst_bw_reg.Q;']
frontend_0.inst_alu_reg = ['frontend_0.inst_alu_reg_11_.Q;', 'frontend_0.inst_alu_reg_1_.Q;', 'fronten
d_0.inst_alu_reg_0_.Q;']
```

Extract all the bits of categorized signals from the data set

```
First difference occurs at clock cycle: 6

Category: frontend_0.pc_reg
Bits from Non-faulty core : frontend_0.pc_reg: [1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 0] F03E
Bits from faulty core : frontend_0.pc_reg: [1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0]     F000

Category: frontend_0.inst_src_bin_reg
Bits from Non-faulty core : frontend_0.inst_src_bin_reg: [0 0 1 1] 3
Bits from faulty core : frontend_0.inst_src_bin_reg: [1 1 0 0]     C

Category: frontend_0.inst_mov
Bits from Non-faulty core : frontend_0.inst_mov: [1]
Bits from faulty core : frontend_0.inst_mov: [0]

Category: frontend_0.inst_jmp_bin_reg
Bits from Non-faulty core : frontend_0.inst_jmp_bin_reg: [0 0 0]  0
Bits from faulty core : frontend_0.inst_jmp_bin_reg: [1 1 1]      7

Category: frontend_0.inst_bw_reg
Bits from Non-faulty core : frontend_0.inst_bw_reg: [1]
Bits from faulty core : frontend_0.inst_bw_reg: [0]

Category: frontend_0.inst_alu_reg
Bits from Non-faulty core : frontend_0.inst_alu_reg: [0 0 0 0 0 0 0 0 0 0 0 0] 000
Bits from faulty core : frontend_0.inst_alu_reg: [1 0 0 0 0 0 0 0 0 0 1 1]     803
```

Tb13 : Core_5

Compare faulty(tb13_core5) with non-faulty core(tb-correct_core1)

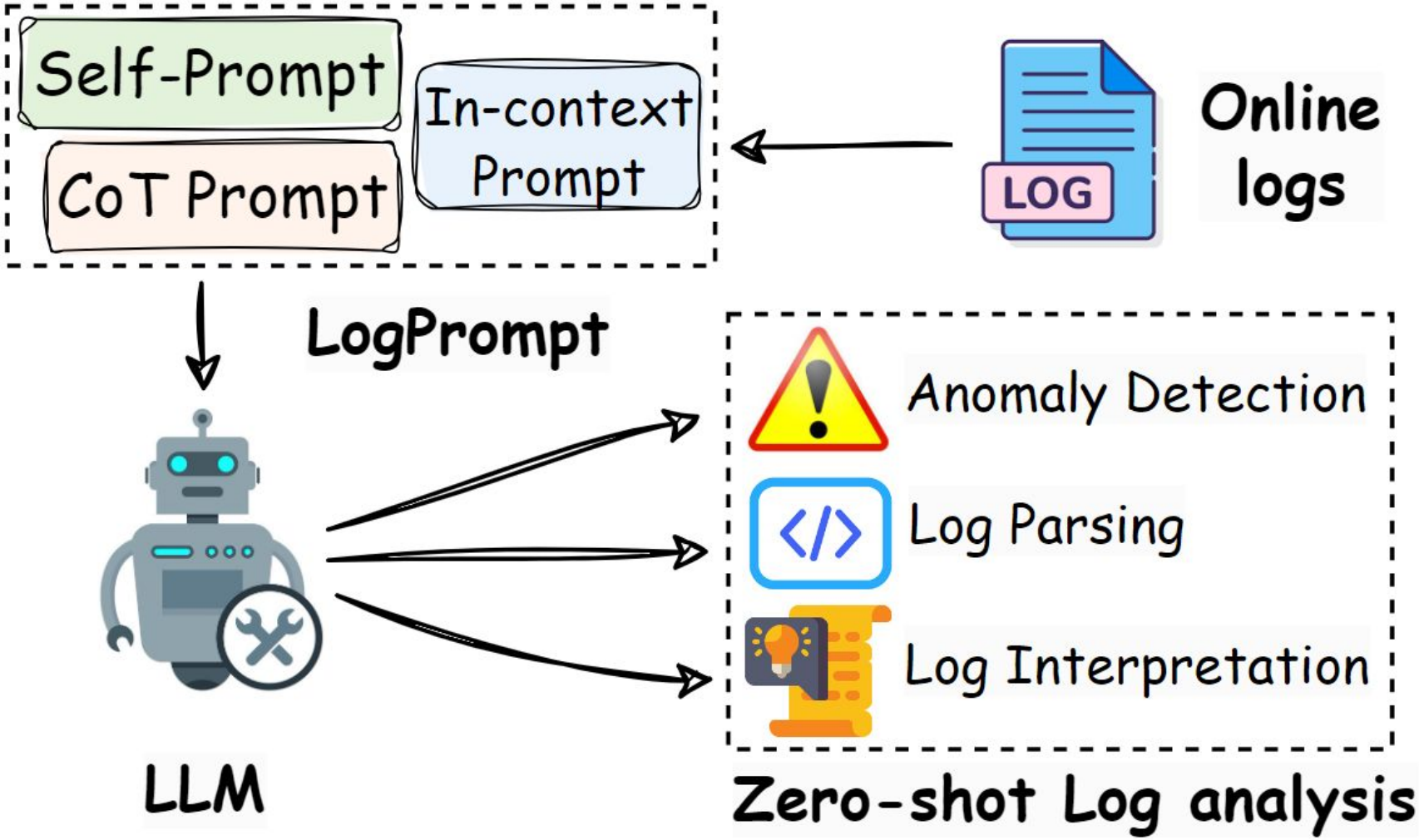


```
First difference occurs at clock cycle: 6

Category: frontend_0.pc_reg
Non-faulty core : frontend_0.pc_reg: [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0] = [F03E]
faulty core : frontend_0.pc_reg: [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0] = [F038]

Category: frontend_0.inst_src_bin_reg
Non-faulty core : frontend_0.inst_src_bin_reg: [0, 0, 1, 1] = [3]
faulty core : frontend_0.inst_src_bin_reg: [1, 1, 0, 0] = [C]

Category: frontend_0.inst_mov
Non-faulty core : frontend_0.inst_mov: [1] = [1]
faulty core : frontend_0.inst_mov: [0] = [0]

Category: frontend_0.inst_jmp_bin_reg
Non-faulty core : frontend_0.inst_jmp_bin_reg: [0, 0, 0] = [0]
faulty core : frontend_0.inst_jmp_bin_reg: [1, 1, 1] = [7]

Category: frontend_0.inst_bw_reg
Non-faulty core : frontend_0.inst_bw_reg: [1] = [1]
faulty core : frontend_0.inst_bw_reg: [0] = [0]

Category: frontend_0.inst_alu_reg
Non-faulty core : frontend_0.inst_alu_reg: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] = [000]
faulty core : frontend_0.inst_alu_reg: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1] = [803]
```

FAULT ROOT CAUSE

CORE5:
openmsp430_0
-frontend_0

pc	F038
inst_src_bin	C
inst_mov_reg	0
inst_jmp_bin	7
inst_bw_reg	0
inst_alu	803


Self-Prompt
In-context Prompt
CoT Prompt
Online logs
LOG
LogPrompt
Anomaly Detection
Log Parsing
Log Interpretation
LLM
Zero-shot Log analysis