ISCAS 2025
IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS
LONDON, UK | May 25-28, 2025

Smart Watchdog Mechanism for Fault Detection in RISC-V

David Simpson, Jim Harkin, Malachy McElholm, Liam McDaid

School of Computing, Engineering and Intelligent Systems

Ulster University – Derry, Northern Ireland, UK

Ulster University

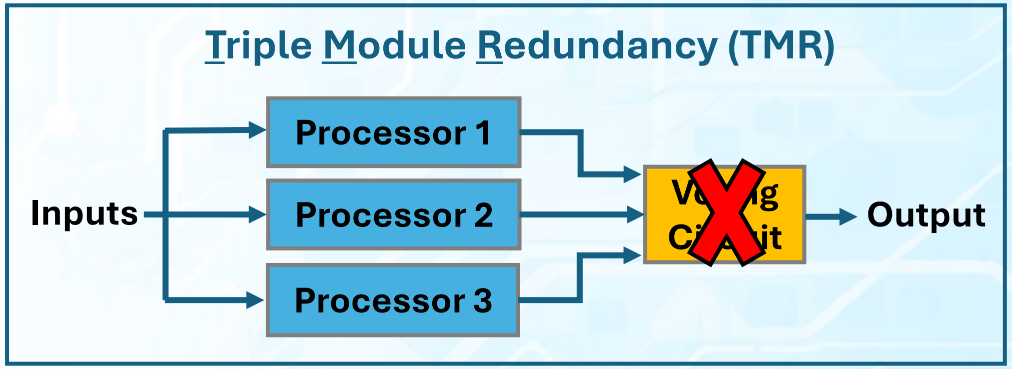
NEORV32

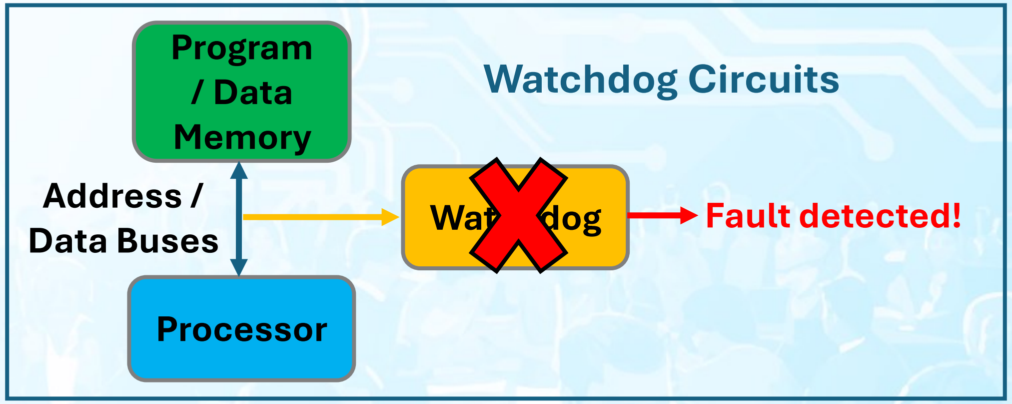
snnTorch

Department for the Economy

Background

- Modern processors are transistor-dense:
  - Increased functionality
  - Decreased reliability
- More prone to hardware faults:
  - Single Event Upsets (SEUs)[1]
  - Safety-critical applications, i.e. space exploration[1]
- Error checking is vital:
  - Hardware redundancy (generic)[1]
  - Watchdog circuits (processor-specific)[2]
- Watchdog design requirements:
  - Minimal area overhead[2]
  - Minimal power consumption[2]
  - Robust to failure[3]

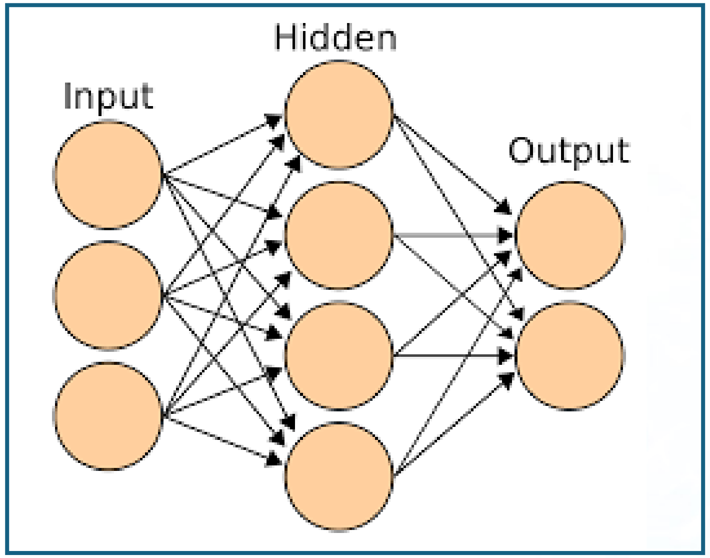




1- E. Wilson et al, "**Neutron Radiation Testing of RISC-V TMR Soft Processors on SRAM-Based FPGAs**"
2 - P. Bernardias et al, "**A new hybrid fault detection technique for systems-on-a-chip**"
3 - T. Ç. Köylü et al, "**Instruction flow-based detectors against fault injection attacks**"

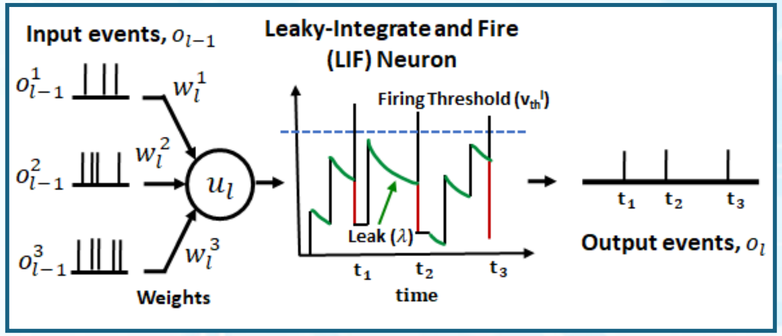
Background (Cont'd)

Artificial Neural Networks (ANNs)



- More reliable structure (dense)
- Efficiency / hardware overheads

Spiking Neural Networks (SNNs)



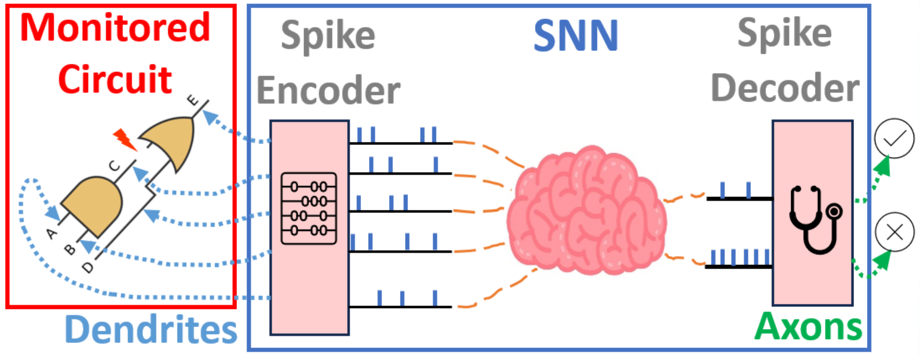
- More efficient* (sparsity - event driven)
- More hardware friendly* (spikes)

**Depending on implementation*

4 - C. Torres-Huitzil et al, "**Fault and Error Tolerance in Neural Networks: A Review**"
5 - M. Dampfhoffer et al, "**Are SNNs Really More Energy-Efficient Than ANNs? an In-Depth Hardware-Aware Study**"

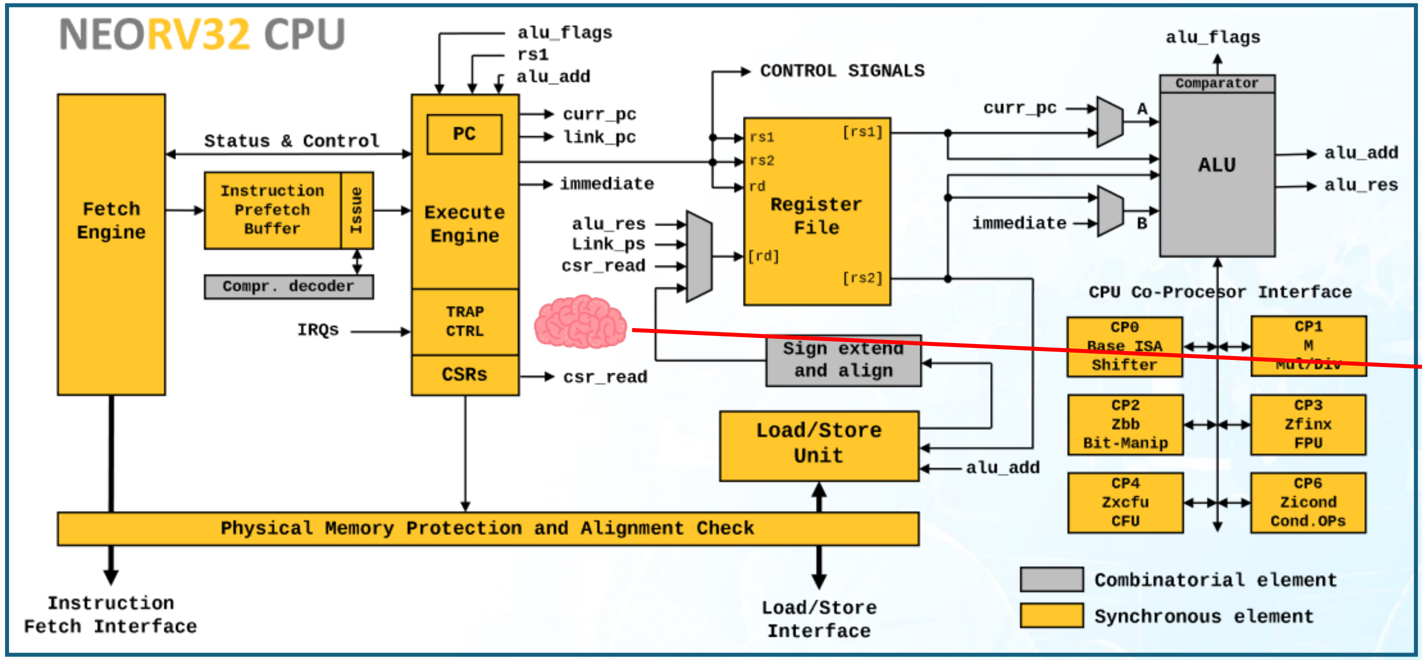
Smart Watchdog

SNN-Based Smart Watchdog



Smart Watchdog Aims:

- ***Smarter*** **watchdog mechanism**
- ***Lower power*** **consumption**
- **More *hardware friendly***

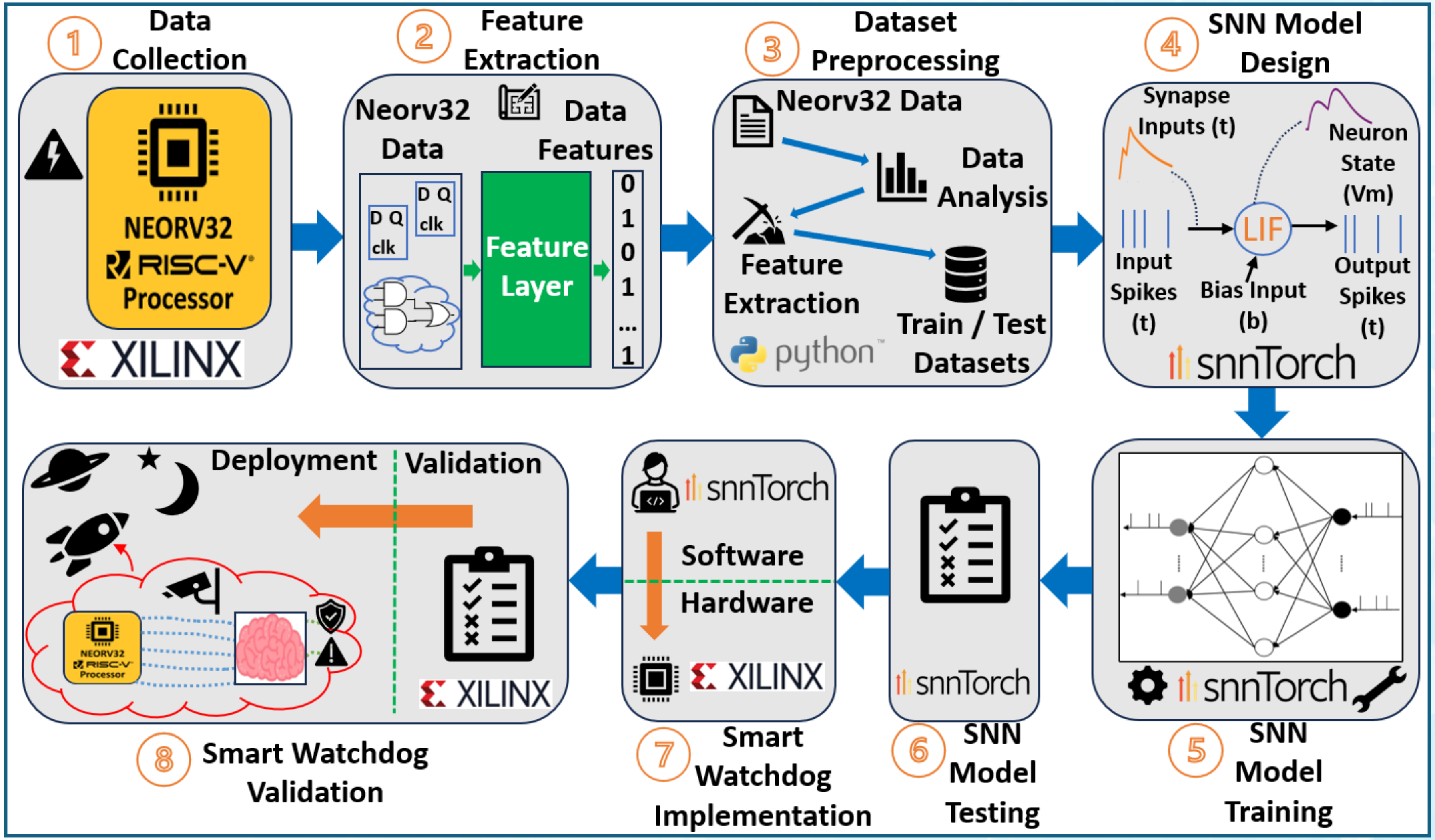


RISC-V Softcore (Neorv32) used as testbed[6]

Smart watchdog trained to monitor control flow at execute stage

6 - S. Nolting, "**The NEORV32 RISC-V Processor - Datasheet**," https://stnolting.github.io/neorv32/.

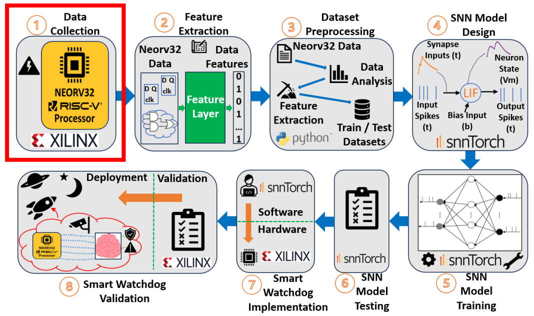
Methodology Overview



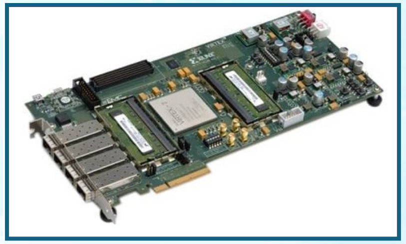
8-Stage Methodology

Methodology

Stage 1 - Data Collection



- High quality data is required to train the SNN.
- Fault injection experiments collected data.



Work carried out on AMD VC-709 (Virtex-7 FPGA)

Data Collection Process

Ran Software Applications on RISC-V

- 3 different software applications:
  - Fibonacci Series
  - Bubble Sort
  - Matrix Multiplication

Injected Faults in Program Counter (PC)

- Fault Models:
  - Bit Flips (SEUs)
  - Temporary stuck at 0/1 (crosstalk, EMI)
  - Permanent stuck at 0/1 (silicon failure)

Extracted RISC-V Data off-FPGA

- Via UART to PC serial terminal
  - RISC-V data stored in text files

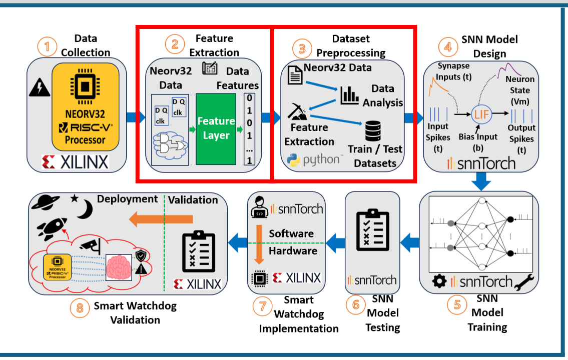
Extracted RISC-V Data

- Program Counter (PC)
- Instruction Register (IR)
- CPU Execute FSM States
- Source Register 1 (RS1)
- Machine Trap Vector Base Address (MTVEC)
- Machine Exception Program Counter (MEPC)

Registers

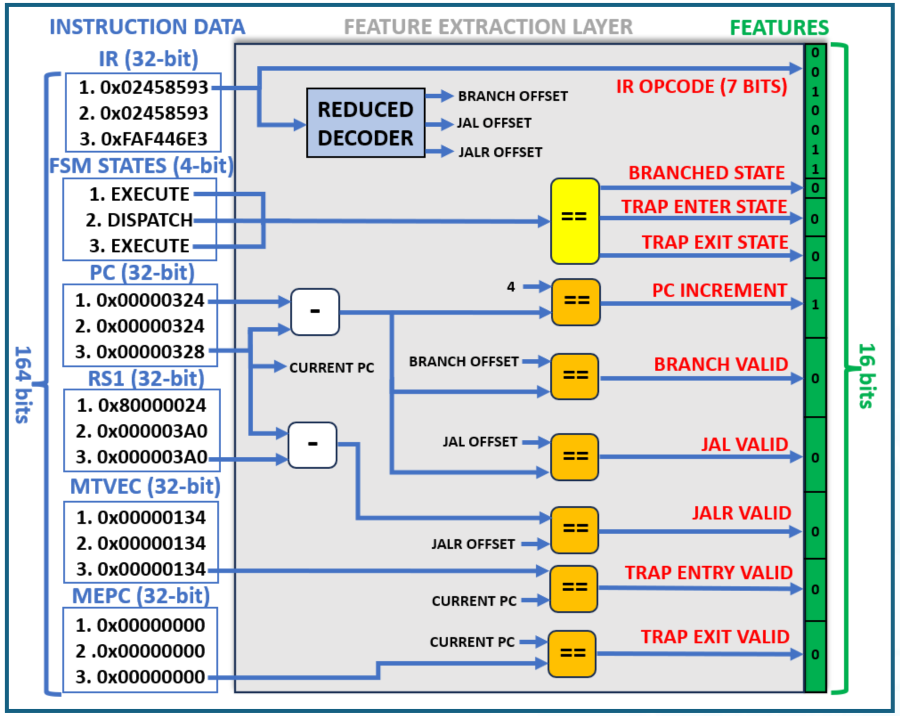
Methodology

Stages 2 and 3 – Feature Extraction and Dataset Preprocessing

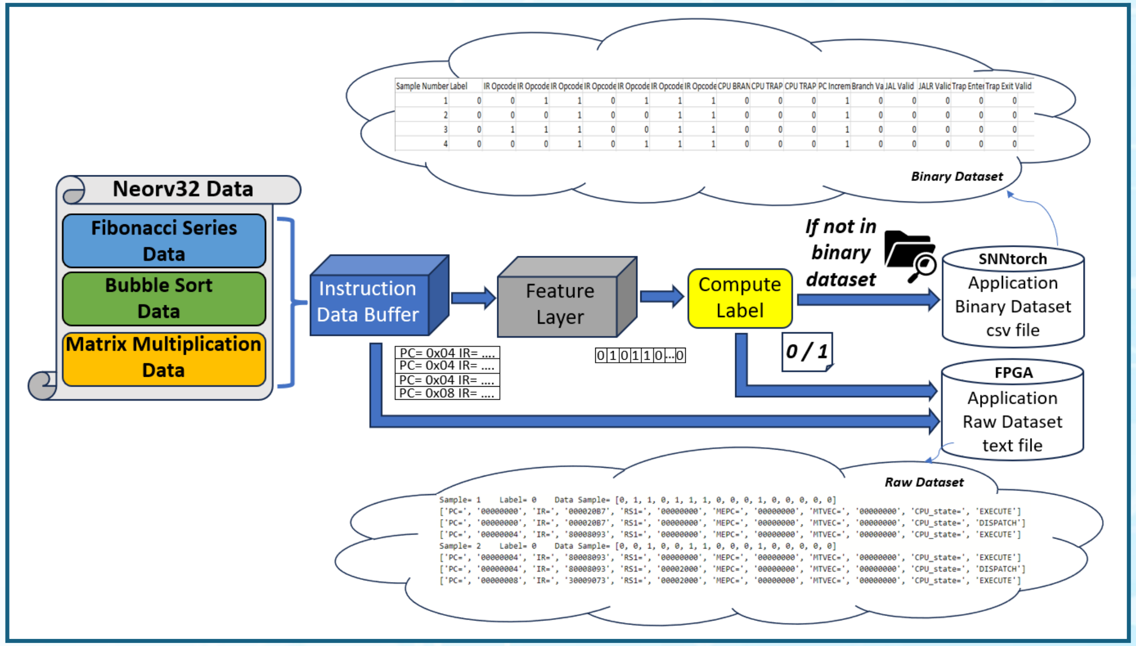


Stage 2 - Feature Extraction

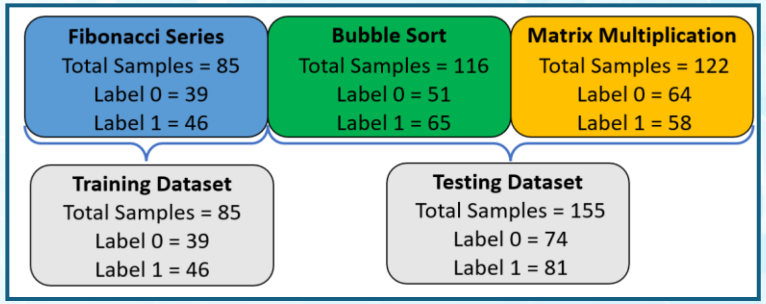
- 16 features captures RISC-V control flow



Stage 3 - Dataset Preprocessing



- ~6.7 million instructions were produced from data collection
- Many instructions produce the same features, e.g. load operations

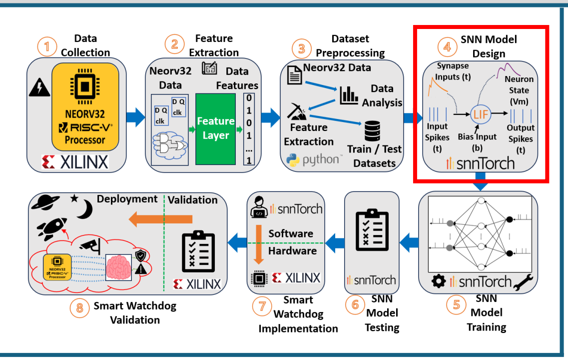


Application Datasets

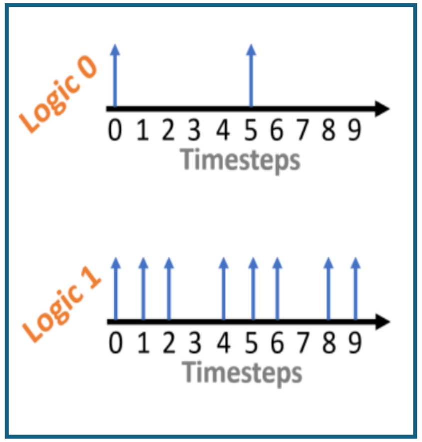
- Unique feature samples

Methodology

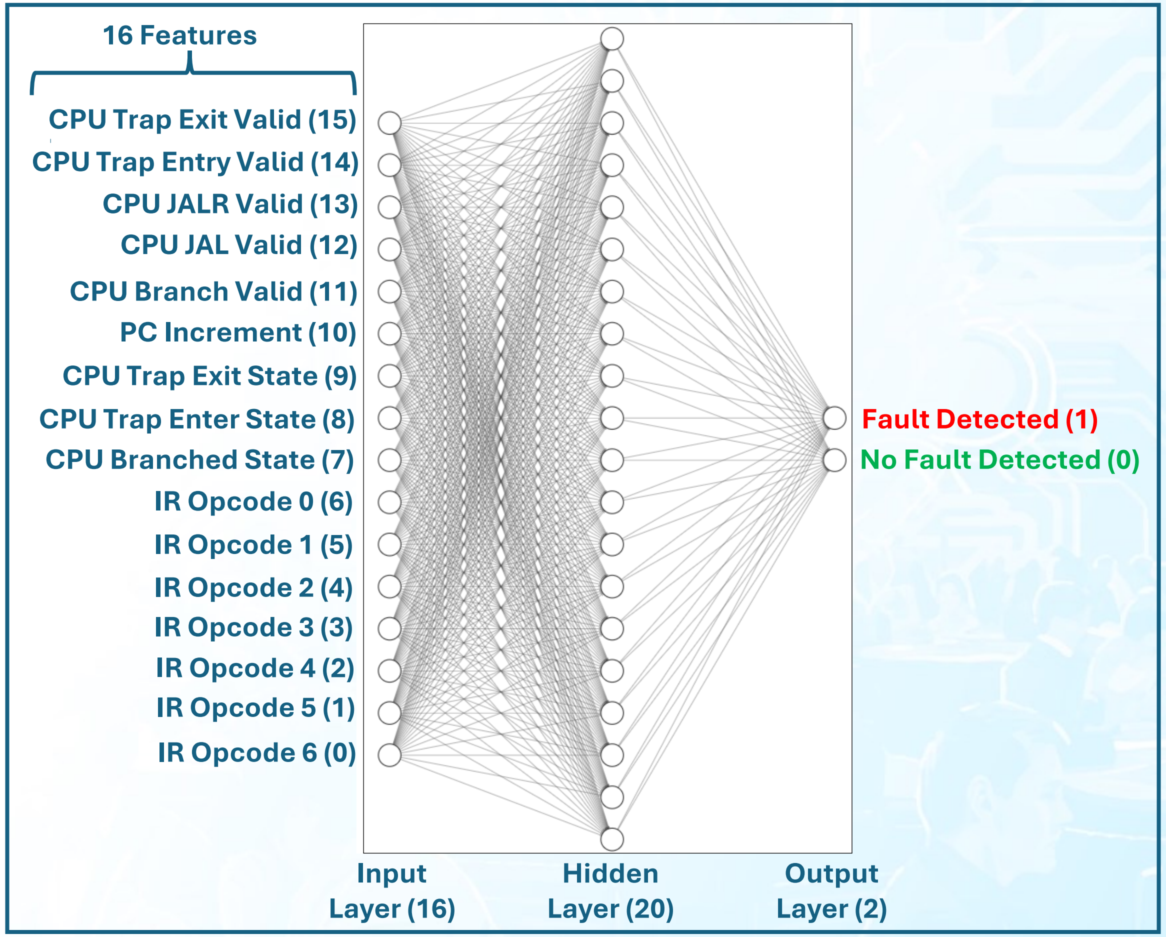
Stage 4 – SNN Model Design



Input Spike Encoding



SNN Model Architecture

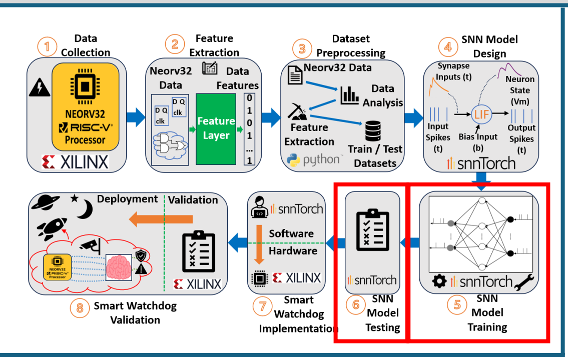


SNN Parameters

Parameter	Value
Network Architecture	Feedforward Fully Connected
Network Size	16 – 20 – 2
Timesteps	10
Layer Biases	Enabled
Neuron Type	1st Order LIF
Neuron Beta	0.75
Resting Potential	0
Threshold	2
Reset Mechanism	Threshold Subtraction
Input Spike Encoding	Custom 2 / 8
Output Spike Decoding	Highest Spike Count

Methodology

Stages 5 and 6 – SNN Model Training and Testing



Stage 5 – SNN Model Training

Training Dataset: Fibonacci Series

Parameter	Value
Network Type	Binary Classifier
Learning Type	Supervised
Development Library	SNNTorch[7,8] (PyTorch)
Epochs	400
Optimizer	Adam
Batch Size	1
Loss Function	Mean Square Error (MSE)
Learning Scheduler Rates	0.1 / 0.01 / 0.001
Learning Rate Milestones	10 / 200

Stage 6 - SNN Model Testing

Negative: normal CPU execution

Positive: control flow error occurred

True Negative / Positive: Smart watchdog classifying **correctly.**

False Negative / Positive: Smart watchdog classifying **incorrectly.**

Testing Dataset: Bubble Sort & Matrix Multiplication

Samples	Correct	TP	TN	FN	FP
155	152	78	74	3	0
Accuracy		**Precision**	**Recall**	**F1 Score**	
0.98		1.00	0.96	0.98	

Seen Test Samples During Training: 80 / 80 (100% accuracy)

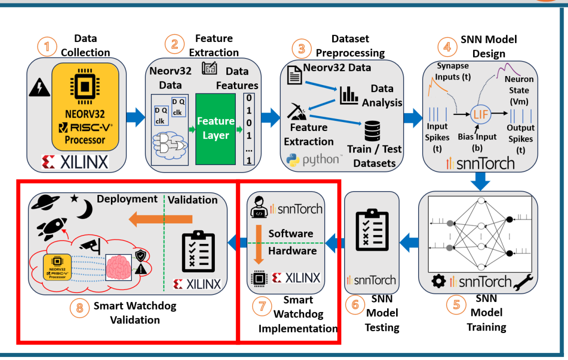
Unseen Test Samples During Training: 72 / 75 (96.0% accuracy)

ISCAS 2025

7 - J. Eshraghian, "**SNNtorch**," https://github.com/jeshraghian/snntorch.
8 - J. K. Eshraghian et al., "**Training Spiking Neural Networks Using Lessons From Deep Learning**"

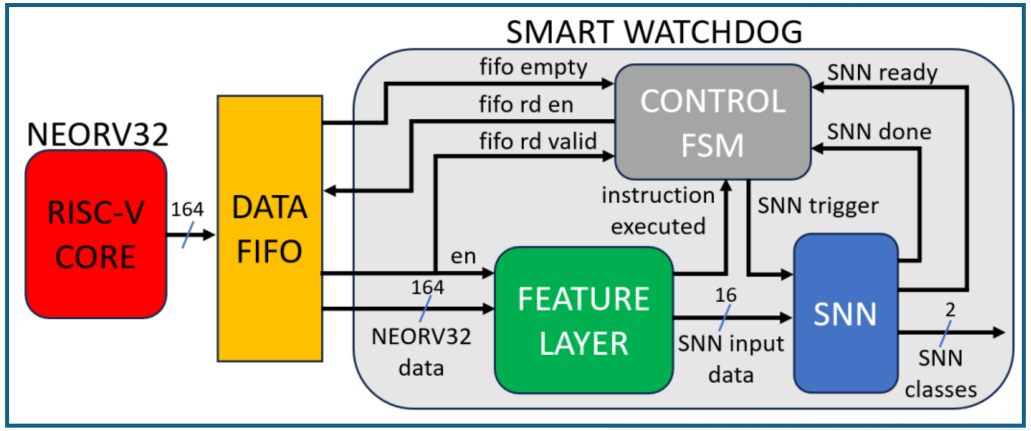
Methodology

Stages 7 & 8 – Smart Watchdog Implementation & Validation



Stage 7 - Hardware Implementation

- 24-bit precision - 10 integer and 14 fractional
- LIF neurons have an adder tree at synapse
- Fmax: 350MHz
- Latency: 153 clock cycles
- Inference Time: ~ 438ns



Hardware Synthesis Results

Component	FFs	LUTs	LUTRAM	BRAM	Power (W)
Neorv32	1,993	1,874	24	6	0.027
Feature Layer	149	157	0	0	0.001
SNN	8,928	6,494	0	0	0.142
Smart Watchdog	10,264	6,733	0	0	0.143

Stage 8 - Hardware Validation

- Heap Sort application for validation (fault injection)
- ~2.4 million instructions were monitored by smart watchdog
- SNN class results extracted (UART) for Python analysis

Validation Dataset: Heap Sort (100 samples)

Samples	Correct	TP	TN	FN	FP
100	98	46	52	2	0

Accuracy	Precision	Recall	F1 Score
0.98	1.00	0.96	0.98

New Samples from Heap Sort: 11 / 11 (100.0% accuracy)

Total Heap Sort Applications	Applications with CFEs	Applications with Traps	Smart Watchdog Detections
1,000	840	490 (58.3%)	350 (100%)

ISCAS 2025

Conclusion

Smart Watchdog Pros

- High fault detection capability (98%).
- Detects control flow errors that the trap handler fails to.
- Can work in tandem with the RISC-V trap handler.
- Requires no modifications for different C code.

Smart Watchdog Drawbacks

- Data collection is required for training the SNN.
- Developed SNN model shows high hardware overheads[9].

Future Work

- Optimize the SNN model:
  - Quantize bit-width
  - Pipelining
- Integrate glial cells (astrocytes) to realise self-repair[10,11].
- Expand smart watchdogs to harden other areas of the RISC-V processor.
- Explore generalization to other processor architectures, e.g. ARM.

9 – Joubert et al, "**Hardware spiking neurons design: Analog or digital?**"
10 - J. Liu et al, "**Exploring Self-Repair in a Coupled Spiking Astrocyte Neural Network**"
11 - J. Liu et al, "**SPANNER: A Self-Repairing Spiking Neural Network Hardware Architecture**"

Questions

Feel free to send any questions to simpson-d12@ulster.ac.uk

LinkedIn:

David Simpson | LinkedIn

Live Demo:

Smart Watchdog Mechanism for Real-Time Fault Detection in RISC-V