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

PhD Supervisors:
Prof Jim Harkin
Mr Malachy McElholm
Prof Liam McDaid

School of Computing, Engineering and Intelligent Systems,
Ulster University, Derry, UK

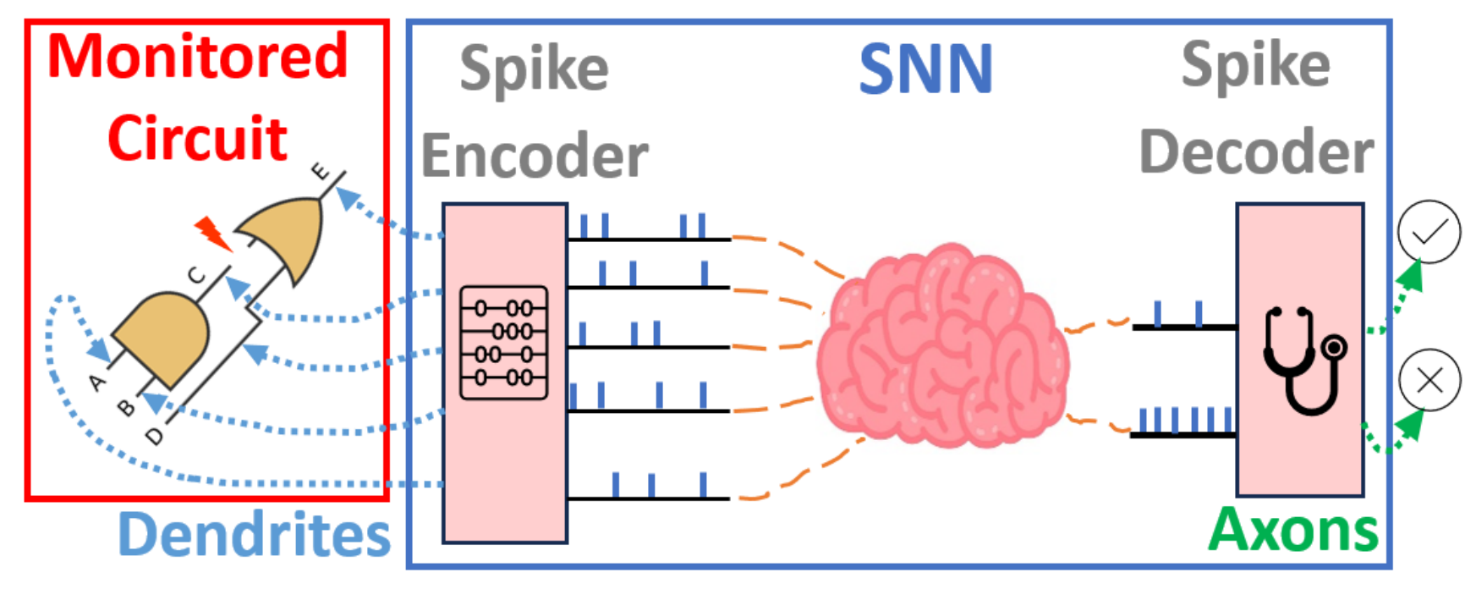
David Simpson
simpson-d12@ulster.ac.uk

Background

- Error checking in modern processors is critical[1].
- Detection mechanisms (watchdogs) must exhibit minimal area overhead and power consumption[2].
- Spiking Neural Networks (SNNs) could offer a more hardware friendly and efficient implementation[3].

SNN-based Smart Watchdog

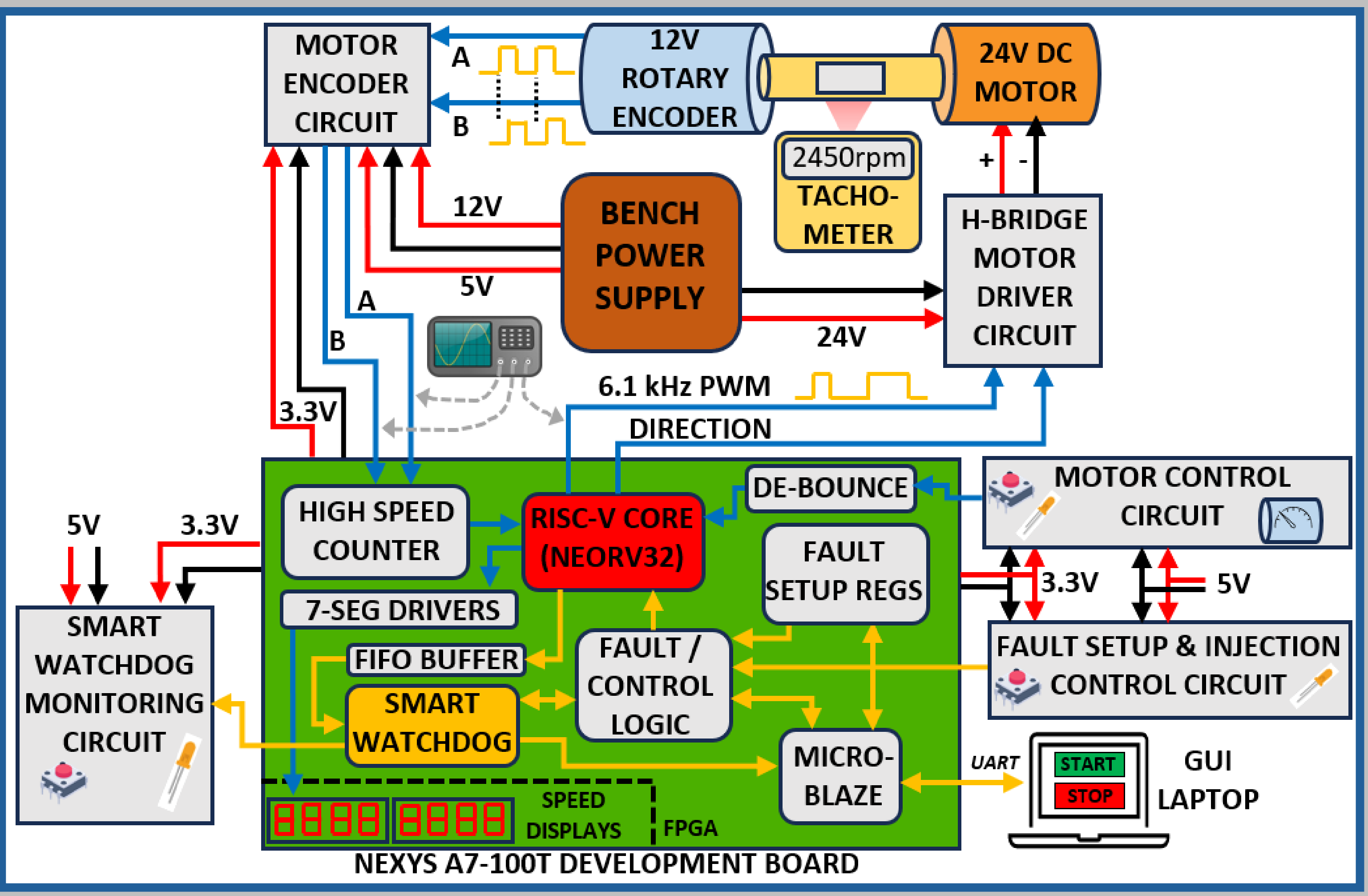
- SNN trained to detect control flow errors (CFEs) caused by hardware faults in a RISC-V processor[4].



Smart Watchdog Aims:
- *Smarter* watchdog paradigm
- *Lower power* consumption
- More *hardware friendly*

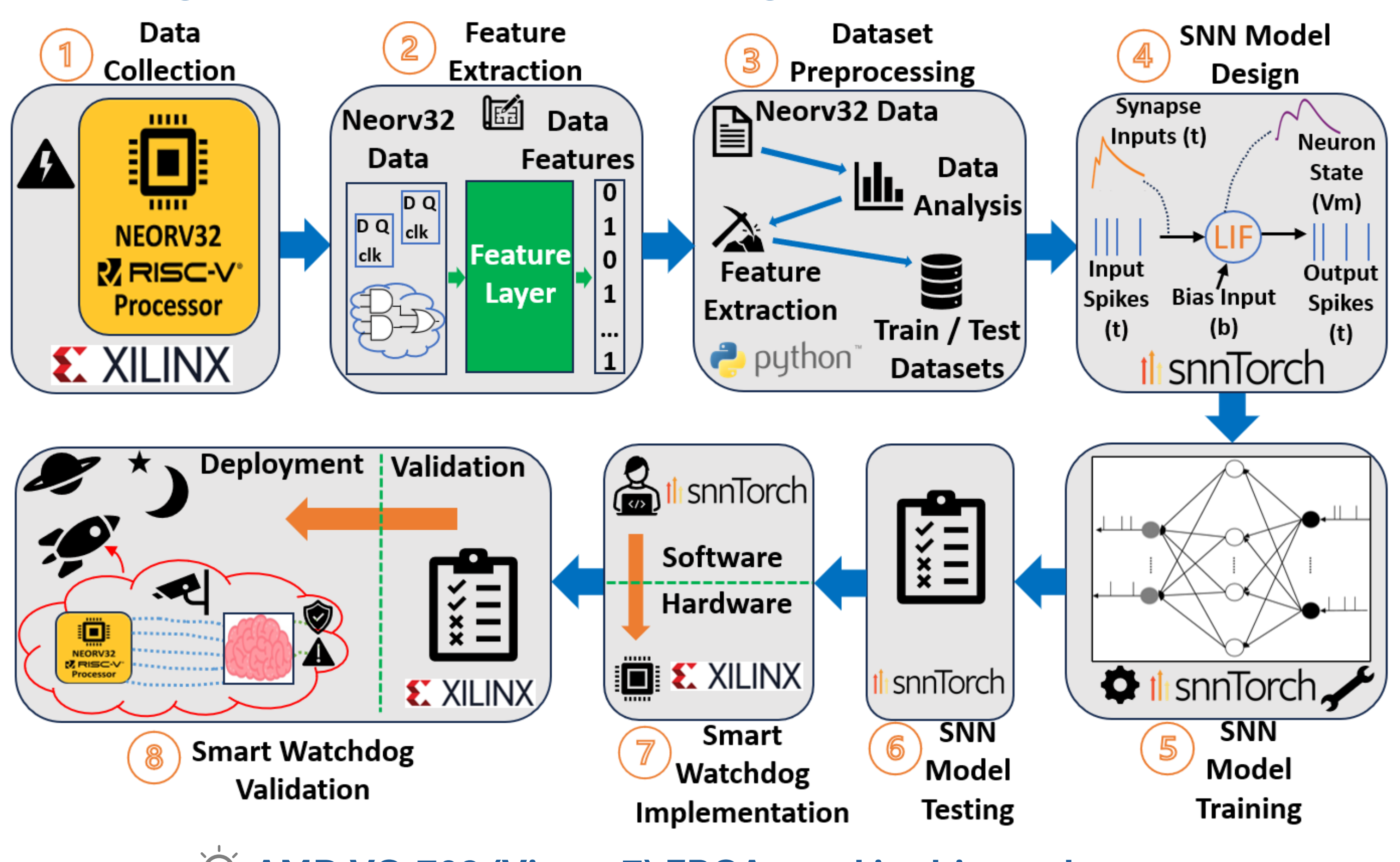
Live Demonstration (FPGA)

- RISC-V processor executes a closed-loop PI control algorithm on a DC motor and rotary quadrature encoder.
- Faults are injected at the program counter register of RISC-V processor.
- Smart watchdog decisions observed in detail on a custom Python-based GUI.
- Nexys A7-100T development board (AMD Artix-7 FPGA) used as hardware platform for live demonstration.
- See Github page for full details.



Methodology

- 8-stages to develop a smart watchdog to monitor a RISC-V processor.



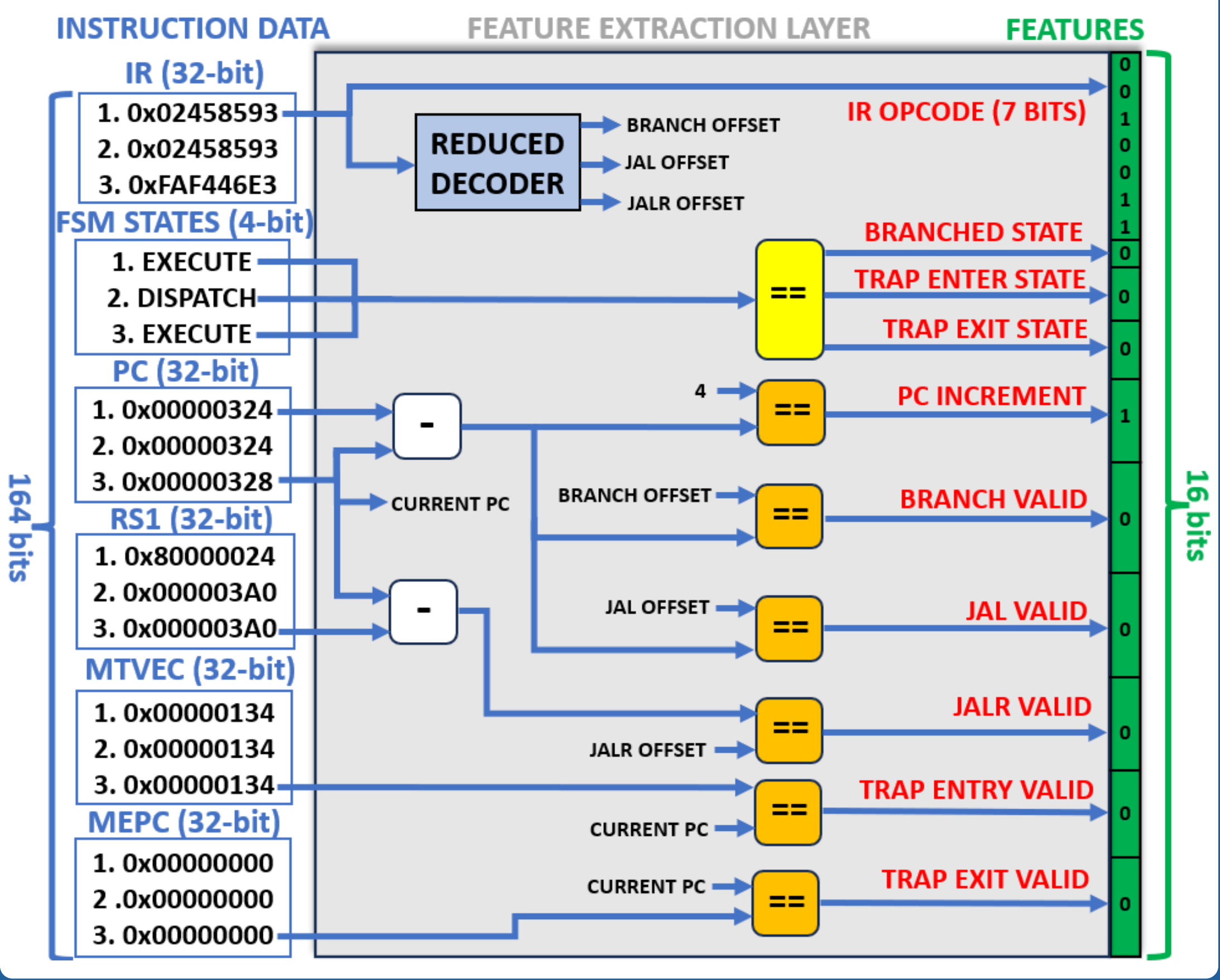
AMD VC-709 (Virtex-7) FPGA used in this work.

Stage 1 – Data Collection

- Fibonacci Series, Bubble Sort and Matrix Multiplication software applications were executed on RISC-V processor.
- Faults were injected into the Program Counter register (PC) of RISC-V processor during execution.
- Instruction data from RISC-V processor was extracted via UART and stored as text files (serial terminal).
- Builds a library of normal instructions and faulty instructions to train the SNN of the smart watchdog.

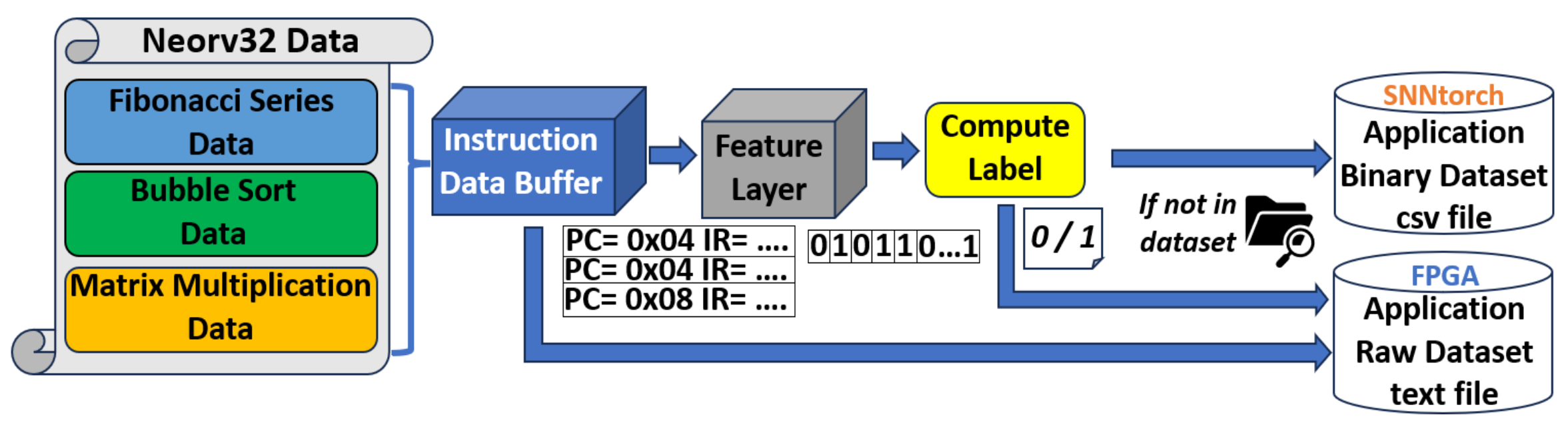
Stage 2 – Feature Extraction

- Features are extracted from each instruction executed, capturing information on control flow.



Stage 3 – Dataset Preprocessing

- Datasets are pre-processed and produced for each of the three applications as shown below.

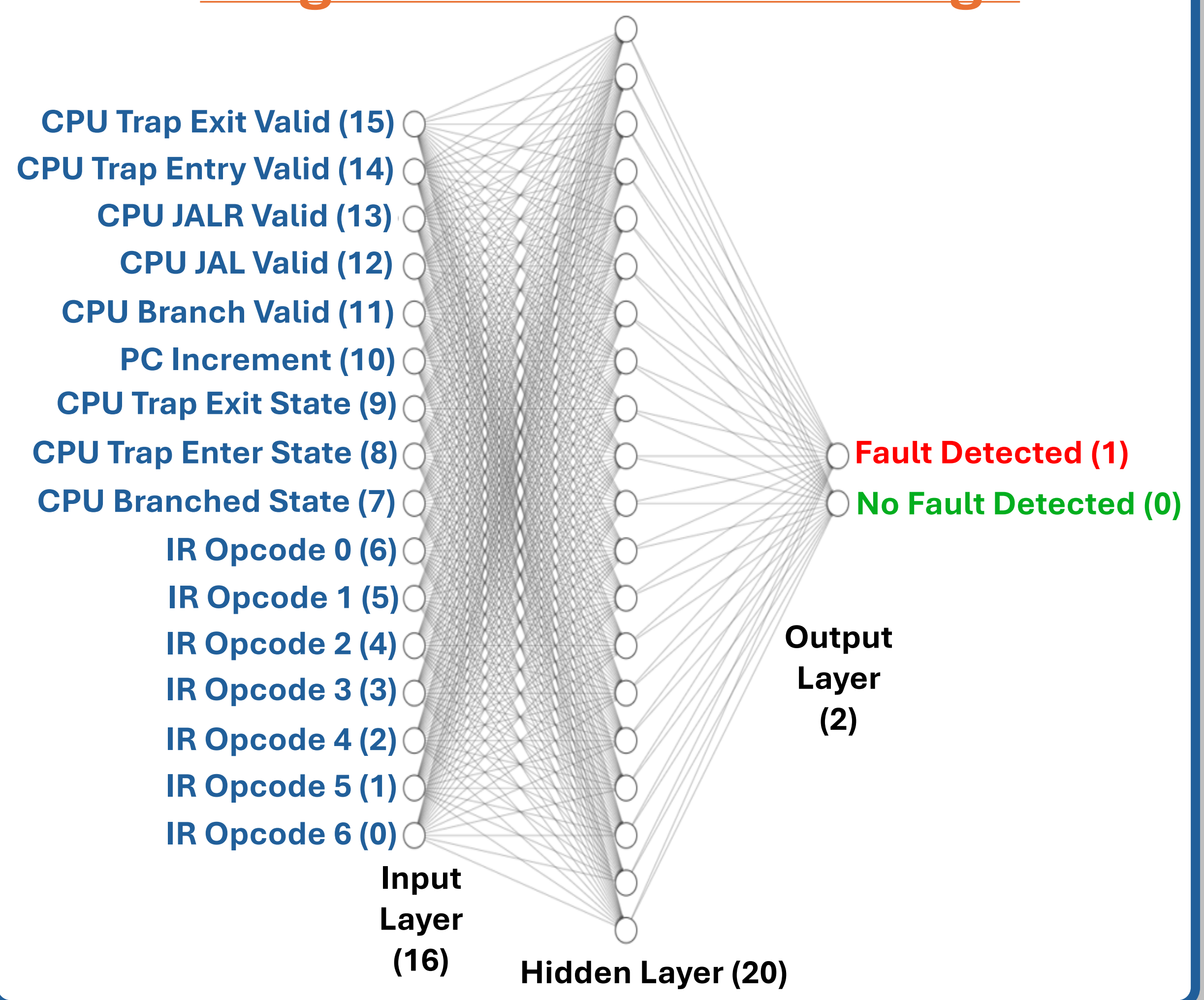


- Training and testing datasets are used to develop the SNN model of the smart watchdog.

Fibonacci Series	Bubble Sort	Matrix Multiplication
Total Samples = 85	Total Samples = 116	Total Samples = 122
Label 0 = 39	Label 0 = 51	Label 0 = 64
Label 1 = 46	Label 1 = 65	Label 1 = 58

Training Dataset	Testing Dataset
Total Samples = 85	Total Samples = 155
Label 0 = 39	Label 0 = 74
Label 1 = 46	Label 1 = 81

Stage 4 – SNN Model Design



Stage 5 – SNN Model Training

Training Dataset: Fibonacci Series

Parameter	Value
Network Type	Binary Classifier
Learning Type	Supervised
Development Library	SNNTorch[5] (PyTorch)
Epochs	400
Optimizer	Adam
Batch Size	1
Loss Function	Mean Square Error
Learning Scheduler Rates	0.1 / 0.01 / 0.001
Learning Rate Milestones	10 / 200

Stage 6 – SNN Model Testing

Testing Dataset: Bubble Sort & Matrix Multiplication

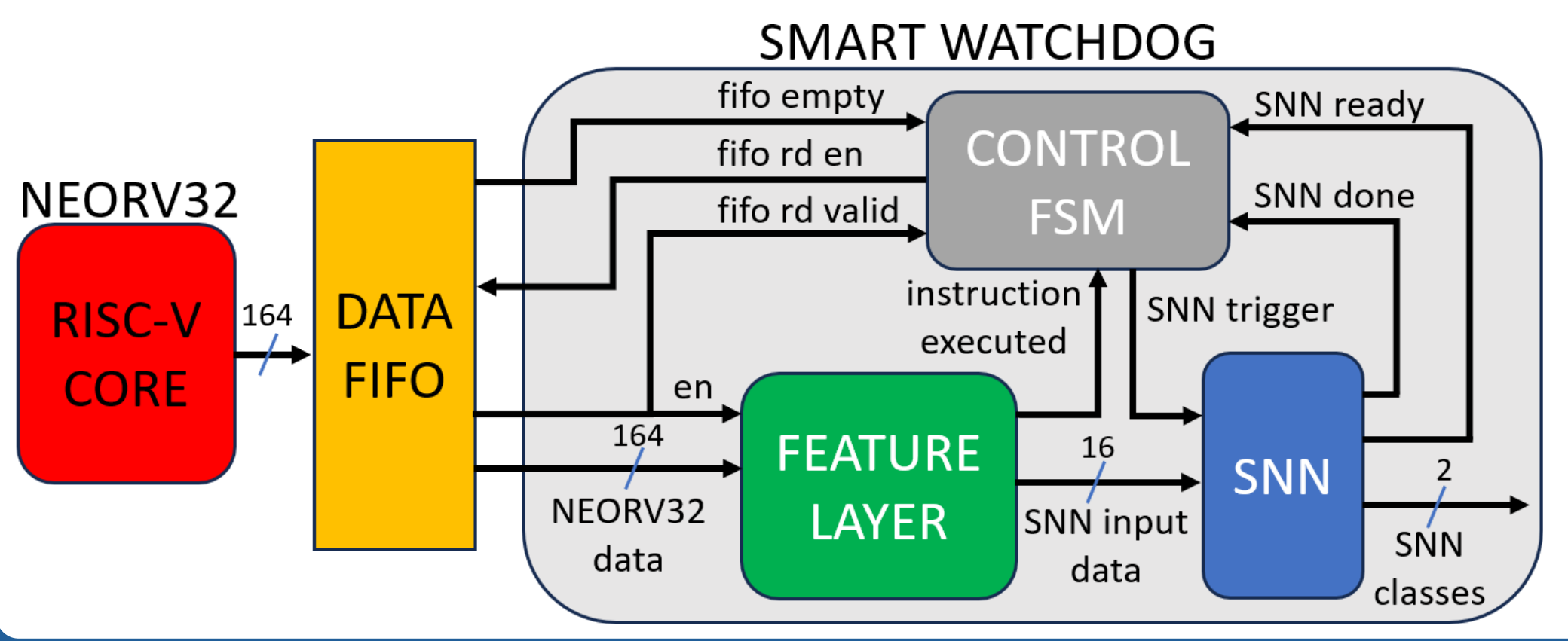
Samples	Correct	TP	TN	FN	FP
155	152	78	74	3	0

Accuracy	Precision	Recall	F1 Score
0.98	1.00	0.96	0.98

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

Unseen Test Samples During Training: 72/75 (96% accuracy)

Stage 7 – Smart Watchdog Implementation



Max Freq: 350MHz

Latency: 153 cycles

Inference: 438ns

Hardware Synthesis Results

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

Stage 8 – Smart Watchdog Validation

Validation Dataset: Heap Sort (new application)

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% accuracy)

Smart Watchdog Fault Detection Capability

Total Heap Sort Applications	Applications with Control Flow Errors	Applications with Traps Triggered	Smart Watchdog Detection
1,000	840	490 (58.3%)	350 (100%)

Smart watchdog detected all CFEs that were undetected in the RISC-V processor.