Journal of VLSI Circuits and Systems, ISSN: 2582-1458 Vol. 7, No. 2, 2025 (pp. 84-98)
WWW.VLSIJOURNAL.COM

RESEARCH ARTICLE

Energy-Efficient VLSI Architecture for Time-Multiplexed Vibration Feature Extraction and Fuzzy Inference

Ahmad Sabiq[1,3], Jazi Eko Istiyanto[1]*, Andi Dharmawan[1], Rachmat Sriwijaya[2]
[1]*Department of Computer Science and Electronics, Faculty of Mathematics and Natural Science, Universitas Gadjah Mada, Yogyakarta, Indonesia.*
[2]*Department of Mechanical and Industrial Engineering, Faculty of Engineering, Universitas Gadjah Mada, Yogyakarta, Indonesia.*
[3]*Department of Informatics Engineering, Universitas YARSI, Central Jakarta, Indonesia.*

KEYWORDS:
Low-Power VLSI,
Integer FIS,
Vibration Feature Extraction,
Time-Multiplexed Architecture,
Low-Power Edge Processing

ARTICLE HISTORY:
Received: 02.12.2025
Revised: 17.12.2025
Accepted: 19.01.2026

DOI:
https://doi.org/10.31838/JVCS/07.02.11

Abstract

This paper presents an energy-efficient VLSI architecture for time-multiplexed vibration feature extraction and fuzzy inference targeted at conveyor gearbox condition monitoring. A triaxial ADXL345 accelerometer mounted on the gearbox housing is sampled at 50 Hz, and 10 integer time-domain features are computed over 1 s windows. Pearson correlation analysis identifies three dominant features—z-axis peak-to-peak, z-axis Willison amplitude, and y-axis Willison amplitude—which form the inputs of a three-class fuzzy inference system (normal, scoring, and damaged). The proposed architecture integrates a digital sensor interface, a two-stage feature-extraction block, and a three-stage fuzzification–inference–defuzzification pipeline using fixed-point arithmetic and extensive time-multiplexing. Two implementations, with eight-bit and four-bit word-lengths, are prototyped on a small low-power programmable logic device. The four-bit variant reduces logic-cell usage from 5105 (66%) to 3550 (46%) and lowers estimated total power from 15.42 mW to 12.46 mW, while maintaining high recall for damaged gears (97% to 96%) and sub-millisecond end-to-end latency per decision. Power figures are obtained from vendor models based on post-route netlists and simulated switching activity, and they are reported together with an effective energy-per-decision estimate to characterize the suitability of the proposed architecture for battery-powered edge deployments.

Authors' e-mail IDs: ahmadsabiq@mail.ugm.ac.id; jazi@ugm.ac.id; andi_dharmawan@ugm.ac.id; sriwijaya@gadjahmada.edu

Authors' ORCID IDs: 0000-0001-7237-8000; 0009-0000-5670-6481; 0000-0002-5780-9869; 0000-0002-7448-0567

How to cite this article: Ahmad Sabiq, et al. Energy-Efficient VLSI Architecture for Time-Multiplexed Vibration Feature Extraction and Fuzzy Inference, Journal of VLSI Circuits and System, Vol. 7, No. 2, 2025 (pp. 84-98).

Introduction

Condition monitoring of geared drives is crucial for predictive maintenance in conveyor-based material-handling systems, where undetected gear faults can cause downtime, productivity loss, and safety risks. Vibration analysis is a well-established approach because tooth wear, scoring, and fractures leave clear signatures in gearbox vibrations. In many plants, however, continuous monitoring must run at the network edge under tight constraints on energy, silicon area, and cost.

Low-power VLSI and small programmable logic devices now enable integrated sensing, signal processing, and decision-making on a single edge node. Recent work has demonstrated low-power system-on-chip architecture for computation-intensive video processing, where adaptive intra prediction, hierarchical motion estimation, and clock gating are used to reduce dynamic power under real-time H.265 encoding constraints.[1] Other designs present fixed-point artificial neural network architecture on FPGA that serve as flexible testbeds for training and inference, optimized for resource utilization and clock frequency and achieving significant speedup over earlier implementations.[2] These examples illustrate how energy-aware architecture can be co-designed with algorithms to meet stringent performance and power budgets.

Beyond these system-level studies, several works implement recurrent or convolutional neural networks on programmable logic for time-series analysis in predictive maintenance, biomedical signal processing, traffic-speed prediction, and inertial or audio tasks.[3-12] These designs show that milliwatt-class devices can support nontrivial time-series models, but they typically assume rich feature sets and focus on generic neural architecture rather than vibration-specific, tightly integrated pipelines.

Fuzzy inference systems (FISs) provide an attractive alternative for condition monitoring thanks to their interpretability, rule-based structure, and natural compatibility with fixed-point arithmetic. Hardware FIS implementations have been reported for power-electronics fault detection, cosmic-ray event triggering, and environmental monitoring, using Mamdani or type-2 TSK formulations. [13-18]. Although these designs achieve microsecond-level latencies and large speedups over CPUs/GPUs, they often target medium- to high-capacity devices and rarely integrate the full chain from digital sensor interface through time-domain feature extraction to fuzzy decision-making, with energy per decision explicitly quantified.

For vibration-based gear monitoring on a tiny fabric, two main tensions dominate. Diagnostic performance benefits from rich features and fine precision, whereas compact hardware demands few integer features and short word-length. Fully parallel pipelines minimize latency but are expensive in logic and switching energy, while time-multiplexing saves both at the cost of internal computation time. This paper addresses these tensions with an end-to-end architecture for conveyor gearboxes—digital accelerometer interfacing, 1 s windowed vibration feature extraction, and fuzzy inference for normal, scoring, and damaged conditions—mapped to a fixed-point, time-multiplexed VLSI data path. Based on an experimental dataset, we select a small set of informative features, implement eight-bit and four-bit variants on a low-power programmable device, and quantify the impact of word-length reduction on power, area, and classification performance.

The main contributions are:

1. **End-to-end edge-oriented model.** A compact vibration-based monitoring model for conveyor gearboxes using three integer time-domain features—z-axis peak-to-peak, z-axis Willison amplitude, and y-axis Willison amplitude—selected via correlation analysis from a broader feature set.
2. **Energy-efficient VLSI architecture.** A time-multiplexed fixed-point architecture that integrates digital sensor acquisition, feature extraction, and a three-stage FIS (fuzzification, rule evaluation, and defuzzification) in a single, low-power VLSI fabric.
3. **Quantized implementations and PPA study.** Eight-bit and four-bit implementations on a small iCE40-class device, with detailed evaluation of resource usage, timing, power, and energy per decision alongside recall for each gear condition.
4. **PPA-accuracy trade-off analysis.** Evidence that the four-bit configuration cuts logic-cell usage by ≈30% and total power by ≈19% versus the eight-bit baseline while preserving high recall for faulty gears, showing that aggressive quantization and time-multiplexing are practical levers for edge condition-monitoring nodes.

Related Work

Hardware acceleration for condition monitoring and time-series analysis has been widely explored. Several works implement recurrent or convolutional neural networks on programmable logic for predictive maintenance, biomedical signal processing, and traffic or inertial data analysis tasks.[4-11,20] A recent design in this direction proposes a 16-bit fixed-point ANN architecture on FPGA that serves as a flexible testbed for training and inference, optimized for resource utilization and clock frequency and achieving significant speedup over prior implementations.[2] These accelerators demonstrate that real-time inference for time-series tasks is feasible on programmable VLSI fabrics, but they typically rely on relatively heavy models and do not integrate the full path from sensor to decision logic.

FISs have also been mapped to reconfigurable and custom hardware for real-time decision-making. Mamdani-type fuzzy triggers have been implemented for cosmic-ray event detection and open-circuit fault detection in cascaded H-bridge inverters,[13,14,21] while more elaborate designs realize interval type-2 TSK FIS engines for wildfire monitoring and multicore FIS accelerators for situation assessment on high-end devices.[15-18] These works achieve microsecond-level latencies and large speedups over CPUs/GPUs, yet generally assume pre-computed features and large device capacities, and they seldom report energy per decision under strict power budgets.

Low-power edge-oriented architecture on tiny programmable devices have been investigated for various sensor and AI workloads. Compact neural networks for capacitive-sensor classification and fall detection have been deployed on small FPGAs[4,5], and quantized GRU/RNN models for livestock-behavior estimation achieve sub-mW to mW power consumption.[9,10,22] Dynamic time warping and LSTM accelerators on ultra-small devices demonstrate that computationally intensive sequence models can run under tight resource constraints, albeit with high logic and memory utilization and board power in the hundreds of milliwatts[3,7,8]. In the video domain, a low-power H.265 SoC implementation leverages adaptive intra prediction, hierarchical motion estimation, and clock gating to reduce dynamic power on a heterogeneous ARM-FPGA platform.[1,23]

Taken together, these works demonstrate that both neural and fuzzy approaches can be efficiently accelerated in hardware, and that tiny FPGAs can support nontrivial time-series models. However, most prior designs either focus on rich neural architecture with substantial resource usage or on FIS cores that assume pre-computed inputs and target larger devices. They rarely integrate the complete chain from digital sensor interface through vibration feature extraction to fuzzy decision-making on a very small fabric, nor do they consistently report energy per decision. In contrast, the present work combines carefully selected integer time-domain features with a compact, time-multiplexed FIS on an iCE40 HX8K, emphasizing short word-lengths and resource reuse to achieve milliwatt-level operation suitable for battery-powered conveyor.

Vibration Dataset and Feature Processing Model

The experimental setup is a belt conveyor driven by a gearbox-motor unit instrumented with a triaxial ADXL345 accelerometer. The sensor is rigidly mounted on the gearbox housing near the output shaft, and its local axes are oriented so that the z-axis captures the dominant radial vibration because of tooth meshing, while the y-axis provides an additional radial component sensitive to misalignment and scoring. Figure 1 illustrates the sensor placement and axis orientation used throughout the experiments.

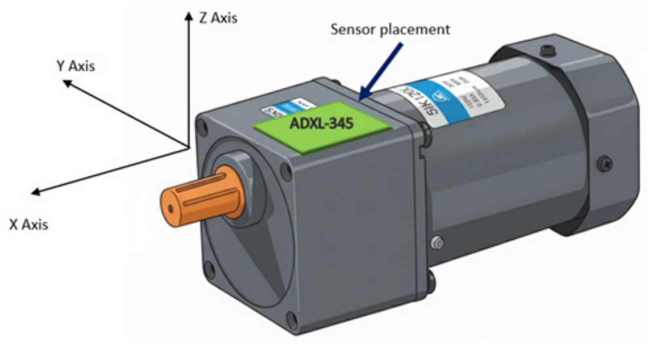

Fig. 1: Placement and orientation of the triaxial ADXL345 accelerometer on the conveyor gearbox housing.

The ADXL345 is configured to a ±2 g range and sampled at 50 Hz via a digital serial interface. The conveyor operates at a shaft speed of 1250 rpm under three gear conditions: normal, scoring, and damaged. For each condition, 1200 s of vibration are acquired, yielding 60 000 samples per axis and 1200 nonoverlapping 1 s windows (50 samples) per class. These windows are split into modelling and testing subsets and form the basis for both fuzzy-model design and hardware validation.

The sampling rate and window length are chosen to balance diagnostic resolution and hardware complexity. At a shaft speed of 1250 rpm, the fundamental mechanical frequency is approximately 20.8 Hz. So, a sampling rate of 50 Hz captures several revolutions per second and suffices for time-domain statistics that are insensitive to high-frequency spectral details. A 1 s window therefore contains about 20–25 shaft rotations and 50 samples per axis, which is long enough to stabilize the time-domain features while keeping the number of samples, and thus the feature-extraction logic, minimal.

For every 1 s window, 10 integer time-domain features are computed from the y- and z-axis signals: Willison amplitude, peak-to-peak value, zero-crossing rate, wavelength, and slope sign change (SSC) for each axis. Feature relevance is quantified using the Pearson correlation coefficient between each feature and the ordinal condition label (normal = 0, scoring = 1, damaged = 2). Table 1 summarizes the correlations and

Table 1: Pearson correlation between integer time-domain features (1 s windows at 50 Hz) and the ordinal gear-condition label (normal = 0, scoring = 1, damaged = 2).

Feature	Pearson correlation
z_willison_amp	0.902992
z_peak_to_peak	0.858746
y_willison_amp	0.781255
y-peak-to-peak	0.657617
z-zero-cross	0.353287
y-zero-cross	0.308598
z-wavelength	0.276626
y-ssc	0.255486
z-ssc	0.141203
y-wavelength	0.110063

shows that three features dominate: z_willison_amp (≈0.90), z_peak_to_peak (≈0.86), and y_willison_amp (≈0.78), while all others are substantially lower. To reduce hardware complexity and avoid weakly informative inputs, the FIS is therefore restricted to these three integer features, scaled to an eight-bit fixed-point range; the four-bit configuration is obtained by truncating the same scaled values.

For model development and evaluation, the 1200 windows per condition are partitioned into disjoint modelling and testing subsets. In the experiments reported here, 70% of the windows (840 per class) are used to design the fuzzy membership functions and to tune the decision thresholds, while the remaining 30% (360 per class) form a held-out test set.

Proposed Architecture

System-Level VLSI Architecture

Figure 2 shows the overall architecture, which implements a complete edge processing chain for conveyor gearbox monitoring. A clock-generation block produces a high-speed core clock (100 MHz) and two divided clocks at 50 Hz and 1 Hz. The digital sensor interface block acquires triaxial acceleration data from the ADXL345 over a serial link at 50 Hz. A feature-processing block computes the three selected time-domain features over 1 s windows. These integer features are fed into a pipelined FIS comprising fuzzification, rule evaluation, and defuzzification stages, which jointly produce a crisp output and a two-bit class label indicating the gear condition.

The architecture is explicitly time-multiplexed: a small set of arithmetic units is reused across streams, features, and fuzzy rules. Within each 1 s window, the low-speed 50 Hz and 1 Hz domains handle data acquisition and window control, while the high-speed core clock services the inner pipelines of fuzzification, inference, and defuzzification. This separation allows the design to meet real-time throughput requirements with modest logic resources and low switching activity.

In terms of hardware cost, the proposed data path is intentionally designed around extensive time-multiplexing rather than full parallelism. A straightforward fully parallel implementation would allocate separate membership-evaluation units for all input-set combinations, distinct MIN/MAX trees for every rule, and dedicated area and moment calculators per output condition, leading to a much larger number of active comparators, adders, and registers. In contrast, the present architecture reuses a single membership core across all three inputs and three fuzzy sets, a shared MIN/MAX core across all rules, and a single area/moment core for all output conditions. This serialization slightly increases internal inference latency but keeps the number of toggling arithmetic units small, which is expected to reduce logic-cell utilization and dynamic power; this effect is quantified later in the experimental results section.

Sensor Interface and SPIcomponent

The SPIcomponent module (Figure 3) provides a reusable and technology-agnostic front-end for digital sensors. It is organized into three hierarchical layers:

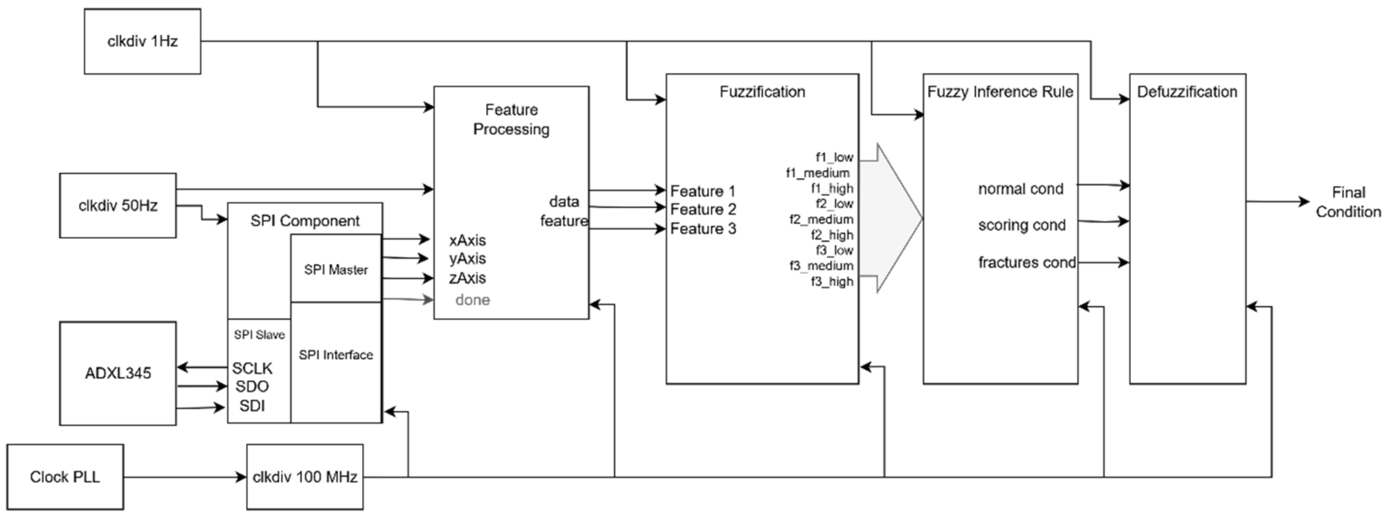

Fig. 2: Overall VLSI architecture integrating digital ADXL345 acquisition, 1 s vibration feature extraction, and the time-multiplexed fuzzy inference system on a low-power programmable device.

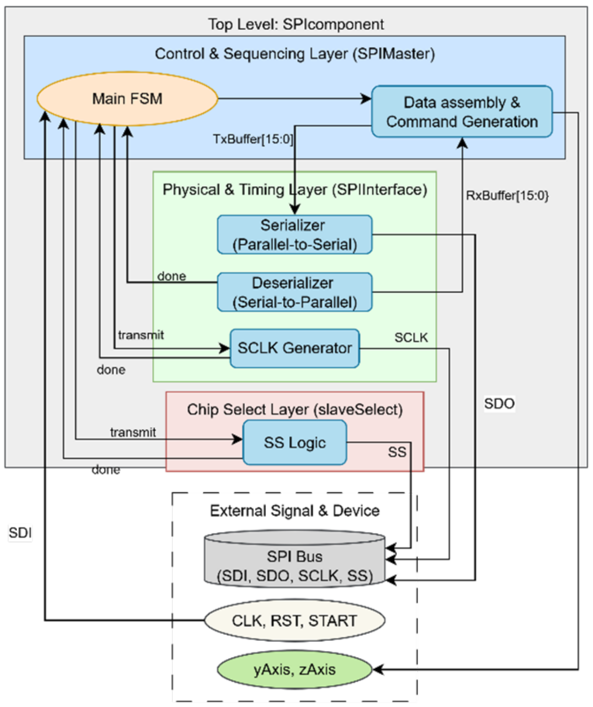
Top Level: SPIcomponent
Control & Sequencing Layer (SPIMaster)
Main FSM
Data assembly & Command Generation
TxBuffer[15:0]
RxBuffer[15:0]
Physical & Timing Layer (SPIInterface)
Serializer (Parallel-to-Serial)
done
Deserializer (Serial-to-Parallel)
transmit
SCLK Generator
SCLK
done
Chip Select Layer (slaveSelect)
transmit
SS Logic
SS
done
SDO
SDI
External Signal & Device
SPI Bus (SDI, SDO, SCLK, SS)
CLK, RST, START
yAxis, zAxis

Fig. 3: SPIcomponent structure: control/sequencing (SPImaster), physical/timing (SPIinterface), and chip-select layer for ADXL345 access.

- A control and sequencing layer (SPImaster) that contains a main finite state machine (FSM) orchestrating all SPI transactions, including register configuration and periodic conversions based on the 50 Hz sampling clock.
- A physical and timing layer (SPIinterface) that hosts the serializer/deserializer pair, the serial clock generator, and the data path for SDI/SDO. Parallel-to-serial and serial-to-parallel conversion are time-multiplexed through shared shift registers to reduce logic.
- A chip-select layer that manages the slave-select signal and encapsulates slave-selection logic, allowing the same SPIcomponent to be extended to multiple devices if required.

Transmit (transmit) and completion (done) handshakes connect these layers: the main FSM triggers a transaction, the physical layer executes the low-level bit transfers, and on completion the received data are placed into an RxBuffer and flagged as valid. The x, y, and z axis samples are then forwarded to the feature processing module without additional buffering, minimizing latency and on-chip storage.

Time-Multiplexed Vibration Feature Extraction

The **data processing** module in Figure 4 implements the three time-domain features using a two-stage architecture: processing_core and output_processing. Inputs y_data and z_data are sampled at clk_50Hz, while clk_1Hz defines the feature window (one second in the present design).

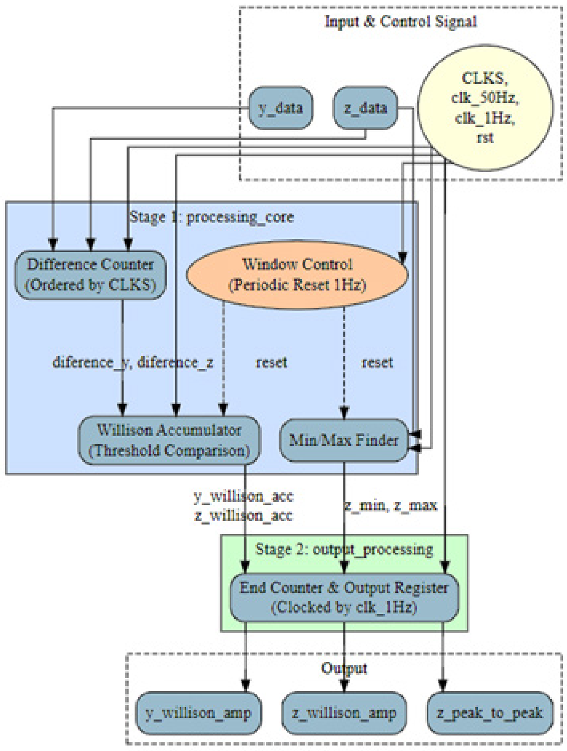
Input & Control Signal
y_data
z_data
CLKS, clk_50Hz, clk_1Hz, rst
Stage 1: processing_core
Difference Counter (Ordered by CLKS)
Window Control (Periodic Reset 1Hz)
diference_y, diference_z
reset
reset
Willison Accumulator (Threshold Comparison)
Min/Max Finder
y_willison_acc
z_willison_acc
z_min, z_max
Stage 2: output_processing
End Counter & Output Register (Clocked by clk_1Hz)
Output
y_willison_amp
z_willison_amp
z_peak_to_peak

Fig. 4: Two-stage feature-extraction block: windowed Willison accumulation and min/max tracking in the processing core, followed by registered feature outputs.

In **Stage 1 - processing_core**, several subblocks operate under a shared control:

- A difference unit computes sample-to-sample differences for the y- and z-axes. The same subtractor hardware is time-multiplexed for both axes.
- A Willison Accumulator integrates the absolute differences that exceed a programmable threshold, yielding y_willison_acc and z_willison_acc. The comparator and accumulator registers are reused for both axes in different clock cycles.
- A Min/Max finder tracks the minimum and maximum values of the z-axis within the current window (z_min, z_max) using a single comparator-update unit.
- A Window Control block, driven by clk_1Hz, generates the periodic reset signals that delimit each feature window and synchronize all accumulators.

Stage 1 thus transforms a stream of raw samples into partial sums and extrema using a compact set of arithmetic units, heavily shared across channels and features.

In **Stage 2 - o**utput_processing, these partial results are latched and converted into final feature values exactly once per window, clocked by clk_1Hz. The Willison accumulators are scaled and truncated to form the integer features y_willison_amp and z_willison_amp. The peak-to-peak feature z_peak_to_peak is computed as the difference between z_max and z_min. Because this stage is active only at the end of each window, it can be clock-gated for the remaining cycles, further reducing dynamic power.

The combination of window-level control and arithmetic reuse ensures that feature extraction meets the 50 Hz sampling requirement while occupying a modest amount of logic and limiting unnecessary switching activity. In a nonmultiplexed feature extractor, the difference calculator, threshold comparators, accumulators, and min/max units would be replicated per axis and per feature, so many arithmetic blocks would toggle in parallel at 50 Hz and at the core clock. In the proposed design, the same subtractor, comparator, and accumulator hardware is reused across the y and z channels under window-level control, so only a small subset of arithmetic units is active in each cycle, which helps contain both logic-cell count and core dynamic power.

Pipelined FIS

The FIS is organized as a three-stage pipeline, as depicted in Figure 5. The FIS receives the three features and associated control signals from the feature-extraction block. A global sequencer coordinates the stages using simple handshake signals: each stage asserts a completion flag, and the subsequent stage is enabled by a corresponding enable signal. This decoupled structure allows each stage to run at an appropriate clock rate and internal latency while maintaining an effective throughput of one decision per feature window.

The pipeline consists of:

1. **Fuzzification stage**, which converts the crisp features into membership degrees for three fuzzy sets (low, medium, and high) per input.
2. **Inference stage**, which evaluates the rule base using MIN-MAX operators to obtain fuzzy support for each output condition (normal, scoring, and damaged).
3. **Defuzzification stage**, which computes a scalar output via a centroid-like operation and maps it to the final class label.

Each stage is internally time-multiplexed, so that only a small number of membership calculators and MIN/MAX operators are required.

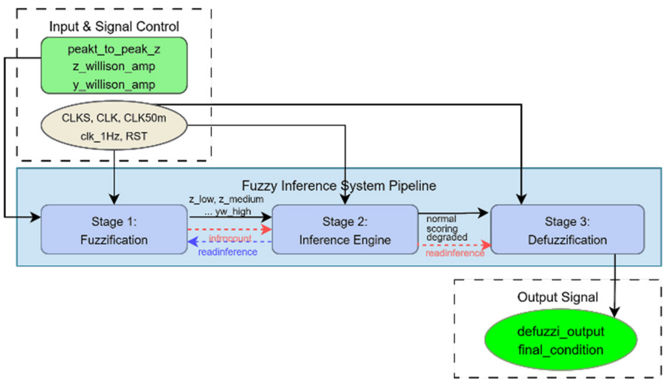

Fig. 5: Three-stage fuzzy inference pipeline (fuzzification, rule evaluation, and defuzzification) with handshake signals between stages.

Fuzzification Engine

The fuzzification module (Figure 6) receives the three feature values together with clock and reset signals. At its core is a main sequencer, driven by a moderate-rate control clock and enabled once per second, which iterates over all input-membership combinations. A small arithmetic unit evaluates trapezoidal membership functions for one feature-set pair at a time.

The membership results are passed to a result multiplexer and intermediate register block, which assembles the full set of degrees (for example, z_low, z_medium, z_high, and analogous sets for the other features) and exposes them to the next stage. A completion signal is raised when all memberships have been computed; the FIS top-level then triggers the inference engine. This scheme reuses a single membership-computation unit for all fuzzy sets and inputs, substantially reducing area compared with a fully parallel implementation.

Inference Engine

The inference module (Figure 7) implements the rule evaluation using a pipelined architecture. A main sequencer generates the sequence of rule evaluations and controls the data flow through the module.

A multiplexer network selects the relevant membership degrees for each rule from the fuzzification outputs. These degrees are fed to a fuzzy operator core, which contains shared MIN and MAX units. For each rule, the core applies the MIN operator to compute the rule firing strength, and then uses MAX operations to aggregate contributions across rules for each output condition.

To reduce storage overhead, intermediate rule strengths are held in a small register bank and are reused via a feedback multiplexer: the same aggregation hardware iterates over the rules instead of maintaining a large

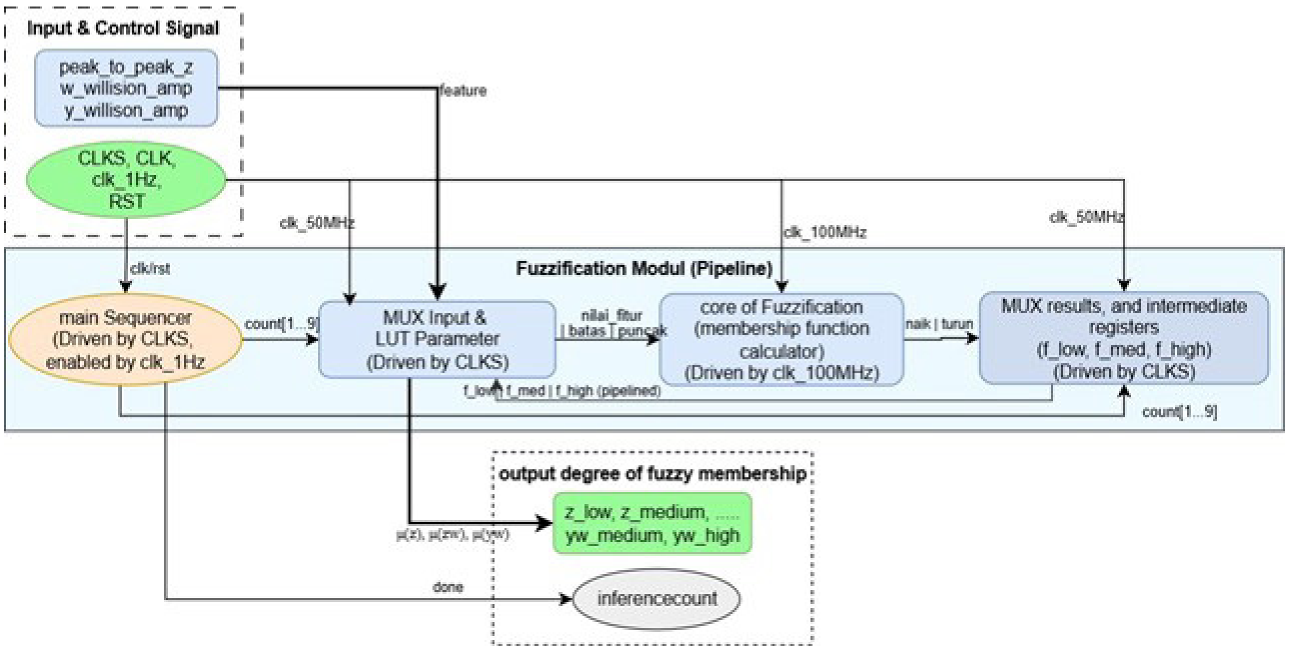
Input & Control Signal
peak_to_peak_z
w_willision_amp
y_willison_amp
feature
CLKS, CLK, clk_1Hz, RST
clk_50MHz
clk_100MHz
clk_50MHz
clk/rst
Fuzzification Modul (Pipeline)
main Sequencer (Driven by CLKS, enabled by clk_1Hz
count[1...9]
MUX Input & LUT Parameter (Driven by CLKS)
nilai_fitur | batas | puncak
core of Fuzzification (membership function calculator) (Driven by clk_100MHz)
naik | turun
MUX results, and intermediate registers (f_low, f_med, f_high) (Driven by CLKS)
f_low | f_med | f_high (pipelined)
count[1...9]
output degree of fuzzy membership
z_low, z_medium, ... yw_medium, yw_high
μ(z), μ(zw), μ(yw)
done
inferencecount

Fig. 6: Fuzzification engine with input and parameter multiplexers, a single membership-calculation core, and staged latching of membership degrees.

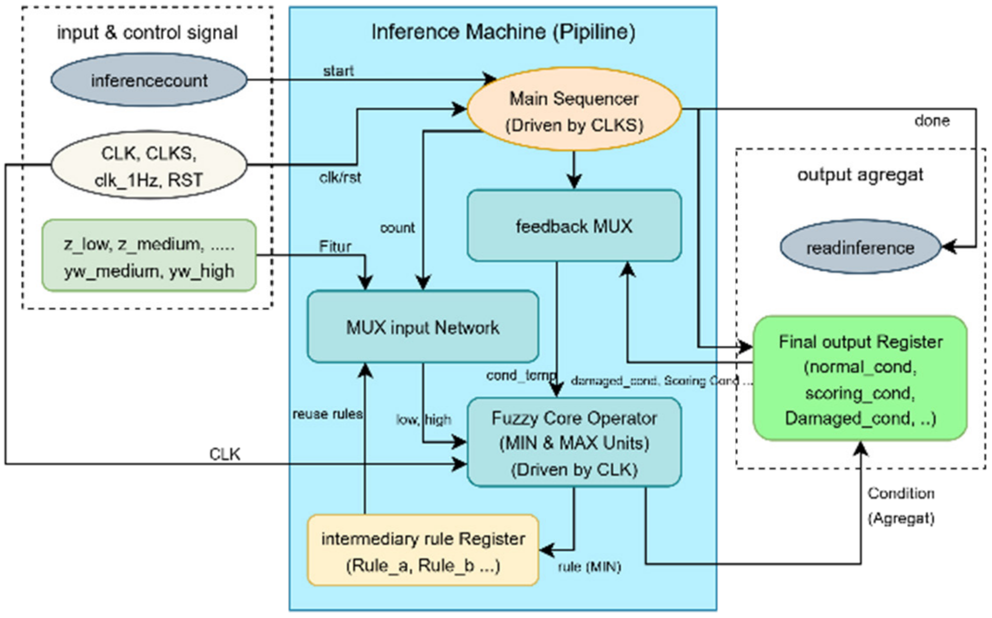
input & control signal
inferencecount
CLK, CLKS, clk_1Hz, RST
z_low, z_medium, ... yw_medium, yw_high
Inference Machine (Pipiline)
start
Main Sequencer (Driven by CLKS)
clk/rst
done
feedback MUX
count
Fitur
output agregat
readinference
MUX input Network
cond_temp
damaged_cond, Scoring Cond...
Final output Register (normal_cond, scoring_cond, Damaged_cond, ..)
reuse rules
low, high
Fuzzy Core Operator (MIN & MAX Units) (Driven by CLK)
CLK
Condition (Agregat)
intermediary rule Register (Rule_a, Rule_b ...)
rule (MIN)

Fig. 7: Inference engine: antecedent MUX network, shared MIN/MAX core, reuse registers, and per-class accumulators.

array of parallel accumulators. When all rules have been processed, the aggregated values for the three output conditions are written into a final output register and passed to the defuzzification stage.

In this design, rule evaluation is explicitly time-multiplexed. A naive fully parallel implementation would instantiate a dedicated MIN/MAX tree—and possibly separate antecedent logic—for every rule and output condition, so that many comparators and adders switch on every core clock edge. Here, the global sequencer instead feeds the required membership degrees to a single shared MIN/MAX operator and a small bank of reuse registers; aggregated supports for each output class are accumulated iteratively via the feedback MUX. This time-multiplexed rule processing reduces the number of MIN/MAX units to a small constant at the cost of a modest increase in inference latency. Since inference is performed only once per feature window and within a budget of hundreds of microseconds, this trade-off is acceptable and beneficial for both resource usage and dynamic power.

Defuzzification Engine

The defuzzification module (Figure 8) receives the aggregated fuzzy supports for the three output conditions, along with a second LUT of triangular output membership parameters. Its pipeline comprises four logical stages under a unified **Sequencer and Control** block:

- An input multiplexer and parameter LUT select one output condition at a time and fetch the corresponding membership parameters.
- An area calculation core, implemented as a multi-cycle arithmetic block, computes in fixed-point the effective membership and the contribution to the numerator for a center-of-area–like aggregation. This core is reused sequentially for each condition.
- An intermediate result register stores the partial membership sums and numerators for each condition. Once all three conditions have been processed, these registers contain the sums required for the final defuzzification.
- A final aggregation and division stage combines the stored values to compute the crisp output defuzzi_output[7:0] and its associated two-bit class label. The division is implemented using an iterative integer algorithm, amortized over the full window period to minimize area.

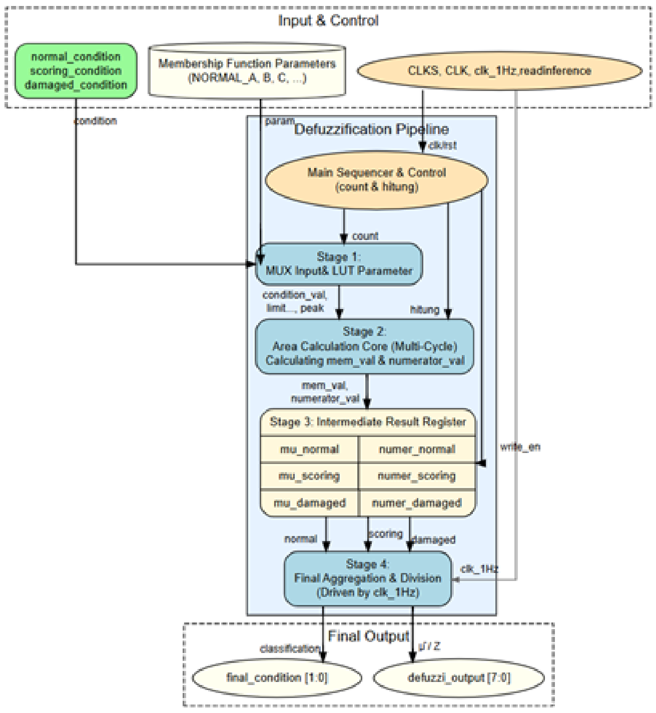

Fig. 8: Defuzzification engine with parameter/load MUX, area and moment core, per-class registers, and final aggregation/division to obtain the crisp output.

The defuzzification pipeline is therefore highly serialized: a single arithmetic core is reused for three output conditions and multiple polygon segments, which substantially decreases logic utilization. Because it is triggered only once per window, the impact on throughput is negligible relative to the 1 s feature window.

Control, Clocks, and Throughput

All blocks described above are coordinated through simple handshake signals and a small number of clock domains. The design uses the 12 MHz on-board oscillator as the primary reference clock. An on-chip PLL multiplies this reference to a 100 MHz core clock that feeds the time-multiplexed data path while two low-frequency clocks at 50 Hz (sensor sampling) and 1 Hz (feature window update) are generated by integer clock dividers from the PLL output.

The high-frequency CLK (100 MHz) is used only in arithmetic cores where multi-cycle computations are beneficial (membership evaluation, MIN/MAX operators, and area calculation). The slower CLKS clock drives the sequencers, MUX networks, and registers that orchestrate time-multiplexing, while clk_1Hz defines the outermost period of feature and inference updates.

The top-level control ensures the following sequence within each 1 s window: (1) continuous sampling of accelerometer data at 50 Hz and accumulation of feature statistics; (2) once the window closes, final feature values are latched; and (3) fuzzification, inference, and defuzzification are executed in sequence, completing well before the next window boundary. Gate-level simulations indicate that the total latency from feature latching to class label is in the order of a few microseconds at 100 MHz, which is orders of magnitude smaller than 1 s and guarantees real-time operation with sufficient timing margin. The multi-clock scheme, combined with time-multiplexed functional units and window-level gating, directly contributes to energy efficiency: only the necessary subcircuits toggle at high frequency for short bursts, while the rest of the architecture remains idle or runs at very low frequency.

Fuzzy Membership Functions and Rule Base

The FIS uses three input variables—z_peak_to_peak, z_willison_amp, and y_willison_amp—and one output variable representing the gear condition. All variables are defined on the eight-bit fixed-point domains obtained from the scaled feature ranges. Each input is partitioned into three trapezoidal sets (low, medium, and high), while the output “condition” employs three triangular sets (normal, scoring, and damaged). The integer breakpoints (a, b, c, and d) for all input and output membership functions are chosen from the empirical feature distributions and snapped to the nearest integers

The rule base follows a Mamdani formulation, but the rules are derived from the labelled dataset rather than specified purely by expert knowledge. For every 1 s window, the three features are fuzzified, and the resulting combination of input labels (low, medium, and high for each feature) is paired with the corresponding ground-truth gear condition (normal, scoring, and damaged). By counting how often each fuzzy-label combination co-occurs with each condition over the entire dataset, dominant patterns are identified: for every distinct fuzzy input pattern, the condition with the highest occurrence frequency is selected as the rule consequent. These initial data-driven rules are then simplified by merging patterns that differ only in one feature but share the same dominant condition, replacing that feature by a wildcard to reduce redundancy while preserving interpretability.

The final rule base contains 14 Mamdani IF-THEN rules that capture the dominant relationships between the three fuzzified features and the gear conditions. The same membership parameters and rule base are used for the floating-point, 16-bit, 8-bit, and 4-bit models; lower word-lengths are obtained by truncating the underlying integer representation, without retraining or retuning the rules. This design choice makes the impact of quantization on both classification accuracy and VLSI cost directly measurable in the experimental results.

Table 2: Integer parameters of input and output fuzzy membership functions (eight-bit configuration).

Variable	Set	a	b	c	d
z_willison_amp	LOW	0	0	5	14
z_willison_amp	MEDIUM	4	12	15	26
z_willison_amp	HIGH	9	21	33	33
y_willison_amp	LOW	0	0	7	15
y_willison_amp	MEDIUM	2	9	12	20
y_willison_amp	HIGH	6	15	27	27
z_peak_to_peak	LOW	9	9	17	27
z_peak_to_peak	MEDIUM	15	24	29	47
z_peak_to_peak	HIGH	24	49	118	118
condition (output)	NORMAL	0	35	98	-
condition (output)	SCORING	22	83	162	-
condition (output)	DAMAGED	55	138	255	-

VLSI Prototype and Implementation Details

The proposed architecture is described in synthesizable Verilog and prototyped on a Lattice iCE40 HX8K device mounted on a small evaluation board. The design instantiates the SPI-based ADXL345 interface, the time-domain feature-extraction block, and the three-stage fuzzy inference pipeline, all mapped to LUT-carry logic without using embedded RAMs or DSP blocks. An on-board 12 MHz oscillator provides the primary clock input; an on-chip PLL multiplies this clock to generate the internal high-speed core clock used by the fuzzy pipeline, while the 50 Hz and 1 Hz clocks for sampling and window control are derived by integer clock dividers from the same source.

Logic synthesis is performed with Yosys, and placement and routing with nextpnr-ice40 under a 100 MHz timing constraint on the core clock. Post-route netlists are used for static timing analysis and for power estimation with the vendor’s iCE40 power models. Switching activity is obtained from gate-level simulations driven by representative vibration traces at 50 Hz. The resource utilization and power figures reported in the following subsections therefore refer to a fully routed design on the HX8K device and correspond to the same operating conditions used in the functional simulations.

Power Evaluation Methodology

The power analysis follows the standard CMOS dynamic-power model in reference[19] and distinguishes between dynamic power in the FPGA core logic, dynamic power in the I/O pins, and static power. The total dynamic power is defined as:

$$P_{\text{dyn}} = P_{\text{core}} + P_{\text{IO}} \tag{1}$$

Core dynamic power is modelled as the product of the average toggle energy and the aggregate logic-event rate:

$$P_{\text{core}} = E_{\text{toggle}} \times \text{Events}/s \tag{2}$$

with the per-transition energy:

$$E_{\text{toggle}} = \frac{1}{2} C_{\text{eff}} \cdot V_{\text{core}}^{2} \tag{3}$$

Here C_{eff} is an effective capacitance that represents the combined load of LUTs, flip-flops, routing, and the clock tree. The aggregate logic-event rate is approximated as:

$$\text{Events}/s = \sum_{k} \left(N_{\text{LC},k}\ \alpha_k\ f_k \right) \tag{4}$$

where $N_{LC,k}$ is the number of active logic-cells in clock domain k, α_k is the average activity factor (probability of a transition per clock cycle) in that domain, and f_k is the corresponding clock frequency.

In practice, $N_{LC,k}$ and f_k are obtained from the post-place-and-route reports (Yosys/nextpnr and iCEcube2), while $\alpha_k f_k$ is taken directly from the *Seq LCs Switching Frequency* and *Comb LCs Switching Frequency* columns of the lattice power estimator. Each entry in these columns already represents the effective switching frequency per logic-cell for that domain, so the total logic-event rate is computed simply as the sum of $N_{LC,k}$ times the corresponding switching frequency. The effective capacitance C_{eff} and toggle energy E_{toggle} are calibrated once by matching Equations (3)-(5) to the core dynamic power reported by the vendor tool for the same post-route netlist and clock configuration; the calibrated E_{toggle} is then reused when comparing the eight-bit and four-bit architecture.

Dynamic I/O power is modelled as the sum of the capacitive switching power over all toggling pins:

$$P_{IO} = \sum_{i=1}^{N} \left(\frac{1}{2} C_{IO,i} V_{IO,i}^2 f_{SW,i} \right) \tag{5}$$

where $C_{IO,i}$ is the effective load capacitance of pin i (including package and board load), $V_{IO,i}$ is the corresponding I/O-bank voltage, and $f_{SW,i}$ is the effective switching frequency derived from the interface specification (e.g., SPI clock) and confirmed by the power estimator. The total device power is then:

$$P_{total} = P_{static} + P_{dyn} \tag{6}$$

Static power P_{static} is taken from the iCE40 HX8K datasheet for the nominal core voltage V_{core} = 1.2 V.

The energy per decision is finally obtained by multiplying P_{total} by the decision period. Since the fuzzy pipeline produces one condition estimate per 1 s vibration window, the average energy per decision is:

$$E_{decision} = P_{total} \cdot T_{decision}, \; T_{decision} = 1s \tag{7}$$

Experimental Results and Discussion

Functional Validation from Simulation

Figure 9 shows the simulation waveform of the SPI-based sensor interface. The 50 Hz sampling clock (clk_50Hz) drives the START signal that triggers a burst transfer on the SPI bus. During each burst, the generated serial clock (SCLK) toggles while SS is asserted to be low, and the sensor data words are shifted out on SDO and captured on SDI. At the end of each transaction, the parallel registers x_axis[9:0], y_axis[9:0], and z_axis[9:0] hold the signed accelerometer samples for the three axes. The traces confirm correct synchronization between SS, SCLK, and the axis registers and demonstrate that the SPIcomponent delivers fresh sensor samples at every 20 ms period as required.

Figure 10 illustrates the subsequent feature extraction for the y- and z-axis data. Over a 1 s interval, the z-axis extrema (z_min, z_max) are tracked continuously, and the peak-to-peak value z_peak_to_peak[7:0] is updated only at the rising edge of clk_1Hz. In parallel, the Willison accumulators y_Willison_acc[15:0] and z_Willison_acc[15:0] integrate the thresholded absolute differences between consecutive samples. At the end of the window, these accumulators are converted into the final integer features y_Willison_amp[5:0] and z_Willison_amp[5:0]. The waveform shows that intermediate accumulator values evolve monotonically within the window and are reset cleanly at each clk_1Hz tick, confirming that the two-stage data processing block generates one consistent triplet of features per second.

Figures 11 and 12 present the internal behavior of the fuzzy inference pipeline for the same input features under eight-bit and four-bit quantization, respectively. In both cases, the fuzzification stage successively produces the membership degrees (z_low, z_medium, z_high, zw_low, ..., yw_high), which then feed the rule evaluation logic to produce aggregated supports for normal_condition, scoring_condition, and damaged_condition. The defuzzification stage accumulates the partial areas (mu_normal, mu_scoring, and mu_damaged), and numerators (numer_normal, numer_scoring, and numer_damaged) compute the denominator, and finally outputs the crisp value defuzzi_output[7:0] together with the class label final_condition[1:0].

As expected, the four-bit configuration exhibits more coarsely quantized membership degrees and area terms, but the evolution of the internal signals remains qualitatively similar and the final class label matches the eight-bit result for all tested windows. These waveforms validate that the time-multiplexed pipeline and multi-stage handshakes operate correctly across the different word-lengths.

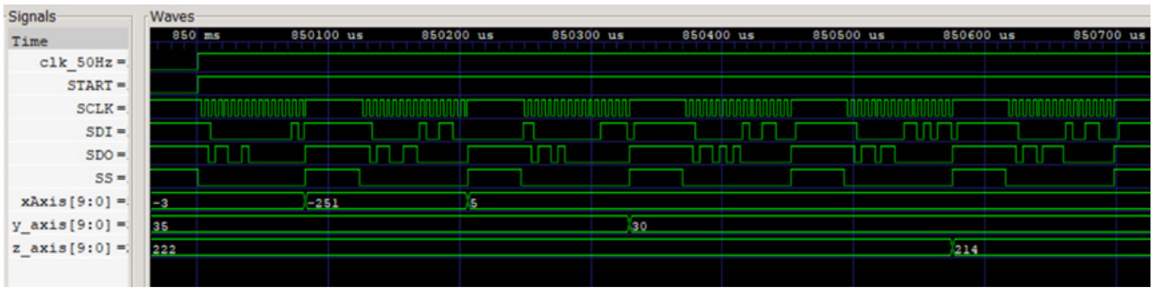

Fig. 9: Gate-level simulation of SPI-based ADXL345 axis-data acquisition at 50 Hz.

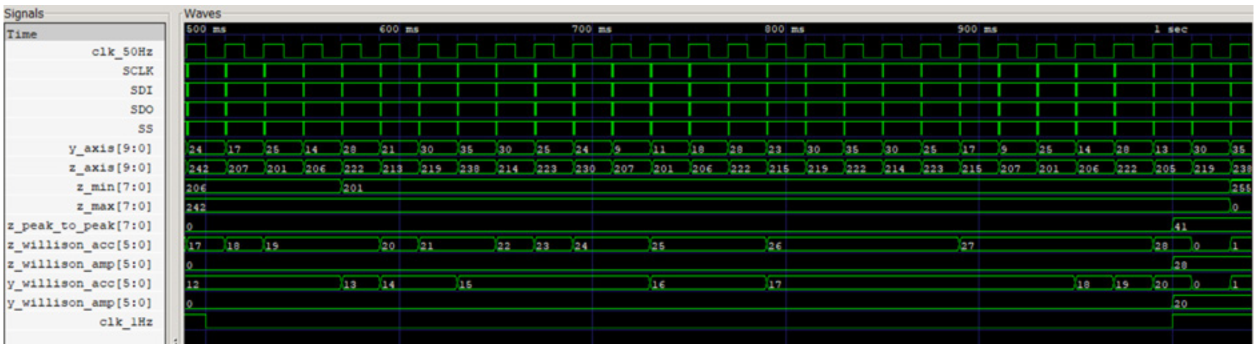

Fig. 10: Simulation of 1 s time-domain feature extraction for the y- and z-axis signals.

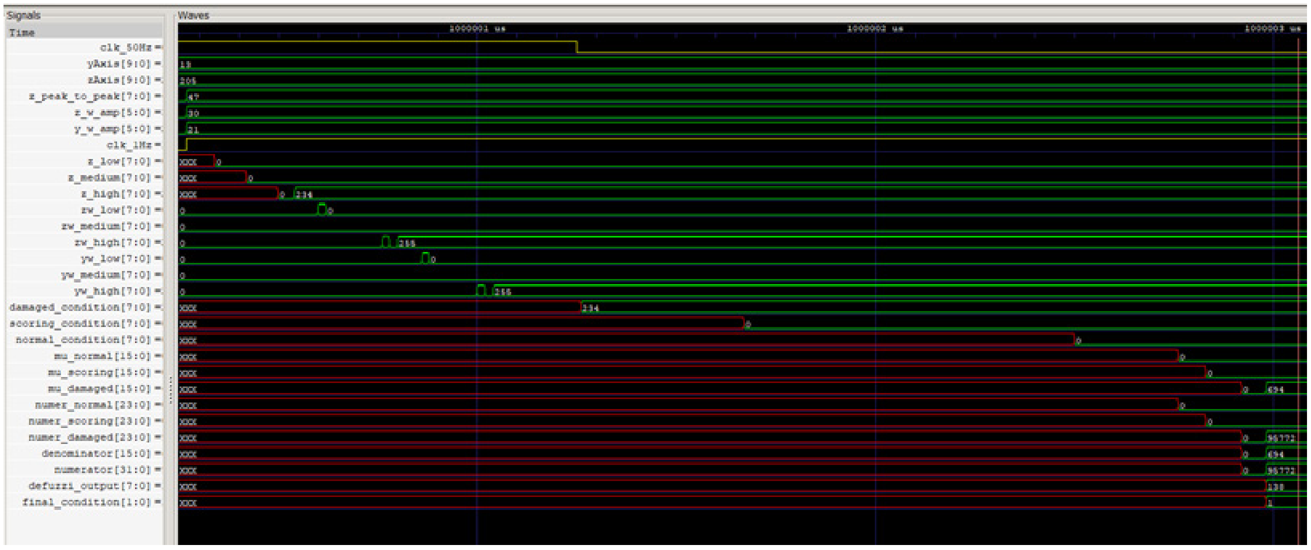

Fig. 11: Internal fuzzy inference signals for the eight-bit quantized implementation.

Resource Utilization on iCE40 HX8K

Table 3 summarizes the post-place-and-route resource utilization of the complete design on the iCE40 HX8K device for the eight-bit and four-bit configurations.

The architecture intentionally avoids dedicated DSP blocks and block RAMs; all arithmetic is realized using LUTs and carry chains, which simplifies portability to ASIC flows. The eight-bit configuration occupies 5105 logic-cells (≈66% of the device), leaving limited headroom for additional on-chip functionality. Reducing the word-length to four bits lowers the logic-cell count to 3550 (≈46%), corresponding to a ≈30% reduction in LUT usage and freeing more than 1500 logic-cells for the integration of communication stacks or higher-level control logic. The I/O and clocking resources are essentially unchanged between the two variants, as they are dictated by the external sensor interface and global clocking scheme.

Both implementations meet the 100 MHz timing constraint on the core clock, with additional slack in the four-bit case because of shorter carry chains and narrower data paths. Since the system issues only one fuzzy

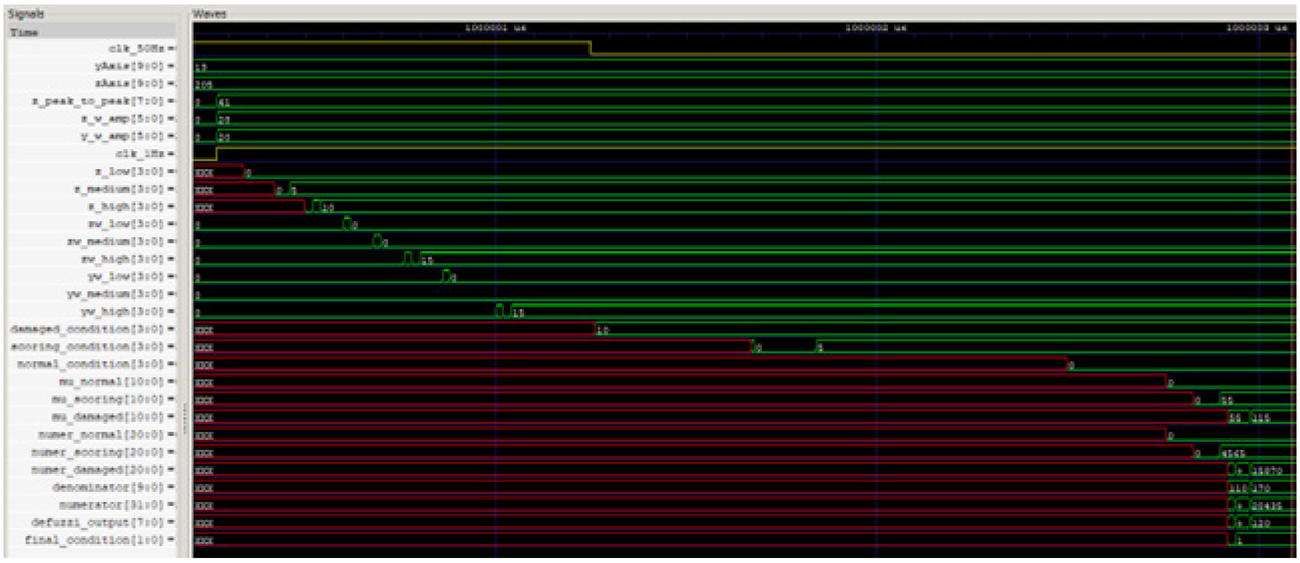

Fig. 12: Internal fuzzy inference signals for the four-bit quantized implementation.

Table 3: Post-place-and-route hardware resource utilization of the complete sensor-feature-FIS pipeline on the iCE40 HX8K device for eight-bit and four-bit configurations.

Resource	Available	Eight-bit usage	Four-bit usage
Logic-Cell	7680	5105 (66%)	3550 (46%)
SB_IO	256	13 (5%)	13 (5%)
SB_GB	8	8 (100%)	8 (100%)
DSP	-	N/A	N/A
PLL	2	1 (50%)	1 (50%)

decision per 1 s window, this additional timing margin does not translate into higher throughput but provides robustness against voltage and temperature variations.

Power and Energy Estimation

The calibrated model above is instantiated using the vendor power data for the eight-bit configuration. In the lattice power estimator, the clock-domain tab for this design contains two active domains: the base clock **main|clk_in** at approximately 7.89 MHz and the PLL core clock **PLLOUTCORE** at approximately 49.34 MHz. For each domain, the tool reports the number of sequential and combinational logic-cells and their *Seq/Comb LCs Switching Frequency* in MHz; these switching frequencies already encode the products $\alpha_k f_k$ in (Equation 4). Summing $N_{LC,k}$ times the corresponding switching frequency over all domains yields a total toggle rate of about $1.16{\times}10^{10}$ events/s for the eight-bit design.

Using the vendor-reported core dynamic power for this configuration, the toggle energy in (Equation 3) is calibrated to approximately:

$$E_{\text{toggle}} = 2.4 \times 10^{-12}\text{J},$$

which corresponds to an effective switched capacitance of about $C_{\text{eff}} \approx 3.35$ pF per event at $V_{\text{core}} = 1.2$ V. The same toggle energy, together with the total logic-cell count and the aggregate $\sum_k \left(N_{\text{LC},k}\ \alpha_k\ f_k\right)$, implies an average activity factor of roughly $a_{\text{eff}} \approx 6.25\%$ for the core logic.

For the comparative analysis, this calibrated E_{toggle} and a_{eff} are then used in Equations (2)-(4) with a single representative operating frequency $f_{\text{clk}} = 12$ MHz, reflecting the oscillator constraint used in the nextpnr timing analysis. The only architectural parameter that changes between the eight-bit and four-bit variants is the number of active logic-cells: $N_{\text{LC}} = 5105$ for the eight-bit design and $N_{\text{LC}} = 3550$ for the four-bit design. This yields:

$$P_{\text{core,8-bit}} \approx 9.23\,\text{mW},$$
$$P_{\text{core,4-bit}} \approx 6.42\,\text{mW}.$$

The I/O dynamic power, computed from Equation 5 using an effective load $C_{\text{load,eff}} \approx 10.6$ pF, an effective switching frequency $f_{\text{SW}} \approx 6.17$ MHz, and 13 active pins at 3.3 V, is approximately:

$$P_{\text{IO}} \approx 4.63\,\text{mW}$$

for both word-lengths, since the external interface and pin count are identical.

Substituting these values into Equations (1)-(6) gives:

$$P_{\text{dyn,8-bit}} \approx 13.86\,\text{mW},$$
$$P_{\text{dyn,8-bit}} \approx 16.27\,\text{mW},$$

Table 4: Classification metrics of the fuzzy inference system for different membership-function quantizations, evaluated on the held-out test set (floating-point and 16-/8-/4-bit integer implementations).

Fuzzy Membership Data Type	Recall Damaged	Recall Scoring	Recall Normal	Macro Recall	Macro Precision	Macro F1-score
Floating-point 64-bit	97	89	63	82.96	86.40	83.06
Integer 16-bit	97	88	64	82.82	85.98	82.64
Integer 8-bit	97	88	64	82.82	85.98	82.64
Integer 4-bit	96	89	64	83.02	86.29	82.86

for the eight-bit architecture, and:

$$P_{\text{dyn,4-bit}} \approx 11.05\,\text{mW},$$
$$P_{\text{dyn,4-bit}} \approx 12.46\,\text{mW},$$

for the four-bit architecture. With one fuzzy decision per 1 s vibration window, the corresponding average energies per decision (Equation 7) are approximately 15.27 mJ and 12.46 mJ, respectively. In relative terms, moving from eight-bit to four-bit quantization reduces the total power and energy per decision by about 19%, in line with the 30% reduction in logic-cell count and the lower toggle rate predicted by the analytical model.

Classification Performance

Table 4 summarizes the classification performance of the FIS for different membership-function quantizations, based on the same rule base and dataset. The 64-bit floating-point model achieves a damaged-gear recall of 97%, with macro recall, precision, and F1-score of 82.96%, 86.40%, and 83.06%, respectively. Moving to 16-bit and 8-bit integer implementations leaves the metrics essentially unchanged, with identical recalls (97/88/64% for damaged/scoring/normal) and very small variations in macro precision and F1-score. The most aggressive four-bit configuration yields a slight reduction in damaged recall to 96%, while improving the scoring recall to 89% and maintaining normal recall at 64%; macro recall and precision remain around 83% and 86%, respectively. These results indicate that the proposed integer FIS is highly robust to quantization, and that the four-bit architecture preserves near-baseline fault-detection performance in spite of its substantially lower area and power.

Discussion

The proposed architecture integrates digital accelerometer interfacing, windowed time-domain feature extraction, and a three-stage FIS into a single, time-multiplexed fixed-point data path on a low-end programmable device. Functional simulations of the SPI-based acquisition, feature-accumulation logic, and fuzzy pipeline confirm that one consistent triplet of features is produced every second and that fuzzification, rule evaluation, and defuzzification complete well within the window period for both eight-bit and four-bit configurations. The internal waveforms for membership degrees, rule activations, and defuzzified outputs remain qualitatively similar across quantization levels, and the resulting class labels are consistent on the tested vibration windows.

The quantitative PPA-accuracy trade-off is summarized by the resource, power, and classification metrics. Reducing the word-length from eight bits to four bits lowers the logic-cell usage from 5105 (66%) to 3550 (46%) of the iCE40 HX8K fabric, a reduction of roughly 30%, while leaving I/O and clocking resources unchanged. At the same time, the estimated total power decreases from 15.27 mW to 12.46 mW, corresponding to about a reduction of 19% in energy per decision at a fixed rate of one decision per second. In terms of diagnostic performance, all integer implementations closely follow the 64-bit floating-point reference. The 16-bit and 8-bit models reproduce the damaged-gear recall of 97% with nearly identical macro recall, precision, and F1-score, and the four-bit configuration reduces damaged recall only slightly (to 96%) while maintaining normal recall at 64% and slightly improving scoring recall. These results indicate that aggressive quantization and extensive time-multiplexing can yield significant savings in area and power with only marginal impact on fault-detection performance.

Although the experiments were conducted on a single gearbox operated at 1250 rpm and nominal load, the proposed model and architecture are not inherently tied to this operating point. The three selected time-domain features (z_peak_to_peak, z-Willison amplitude, and y-Willison amplitude) remain meaningful for other speeds and loads as long as the sampling rate captures several shaft revolutions per analysis window. In practice, adapting the system to different regimes would involve reconfiguring the sampling and window dividers (e.g., to change the number of samples per revolution), re-scaling the fuzzy membership breakpoints, and regenerating the rule base from data collected at the new operating conditions. These modifications affect only the parameter values, not the micro-architecture: the same time-multiplexed data path, resource footprint, and energy per decision are preserved, while the achievable diagnostic accuracy becomes a function of how well the fuzzy model is reidentified for each speed/load scenario.

Compared with representative VLSI and FPGA implementations for fuzzy inference and condition monitoring, the proposed design differs in that it integrates sensor interfacing, time-domain feature extraction, and the FIS pipeline on a single low-power fabric, and reports energy-per-decision under realistic operating conditions. Many existing works focus solely on the inference core and assume precomputed features or target mid-range devices and higher clock frequencies with power in the tens to hundreds of milliwatts and without a consistent per-decision metric. In contrast, the present four-bit implementation delivers complete end-to-end processing within approximately 12–15 mW on a very small device, with a documented trade-off between quantization level, recall, and resource usage. In spite of heterogeneous reporting conventions in the literature, the results here indicate that a carefully time-multiplexed and quantized architecture can provide competitive fault-detection accuracy while significantly reducing area and energy compared with more conventional, parallel, or floating-point designs.

There are, however, several limitations to the present study. First, all power and energy figures are derived from vendor power models and simulated switching activity rather than direct board-level measurements; so, the absolute numbers should be interpreted with caution. Second, the vibration dataset corresponds to a single gearbox and a single operating point (1250 rpm, nominal load). While the hardware pipeline itself is agnostic to speed and torque, the current membership functions and rule base are tuned only for this regime; a systematic sensitivity study across multiple speeds, load profiles, and fault severities is left as future work before deployment in heterogeneous conveyor lines. Third, the prototype targets a specific low-end programmable device; while the architectural principles of time-multiplexing and short word-lengths are directly transferable to ASIC implementations, a full ASIC design and silicon measurements are beyond the scope of this paper.

Conclusion

This paper has presented a compact VLSI architecture that integrates digital accelerometer interfacing, 1 s windowed vibration feature extraction, and a three-stage FIS for conveyor gearbox condition monitoring. By selecting three highly correlated integer features and mapping the entire pipeline to a time-multiplexed fixed-point data path, the design fits comfortably within a low-end programmable device while preserving real-time operation. The eight-bit implementation already achieves microsecond-scale fuzzy inference latency and high recall for damaged gears, and the four-bit variant further reduces logic-cell usage by about 30% and total power by about 19% with only marginal degradation in diagnostic performance.

Future work will extend the approach along several directions. First, the feature set and FIS will be adapted to variable conveyor speeds and additional sensing modalities such as motor current, temperature, or acoustic emissions. Second, optimization techniques such as evolutionary tuning of membership functions and hybrid fuzzy-neural models will be explored under strict word-length and resource constraints. Finally, the architecture will be migrated to more recent low-power VLSI platforms, including newer programmable fabrics or custom ASIC implementations with integrated nonvolatile memory and wireless communication, enabling scalable deployment of energy-aware condition-monitoring nodes in industrial conveyor lines.

Acknowledgments

This work was partially supported by the Department of Computer Science and Electronics, Universitas Gadjah Mada under the Publication Funding Year 2026. In addition, the authors express their gratitude for the valuable comments and suggestions provided by the reviewers, contributing to the improvement of the presentation of this research.

References

1. Naidu, T. M. P., Sekhar, C., & Boya, P. K. (2024). Low power system on chip implementation of adaptive intra frame and hierarchical motion estimation in H.265. Journal of VLSI Circuits and Systems, 6, 40-52. https://vlsijournal.com/index.php/vlsi/article/view/123
2. Arora, G. (2023). Design of VLSI architecture for a flexible testbed of artificial neural network for training and testing on FPGA. Journal of VLSI Circuits and Systems, 6, 30-35. https://doi.org/10.31838/jvcs/06.01.05

3. Abdullah, D. (2024). Design and implementation of secure VLSI architectures for cryptographic applications. Journal of Integrated VLSI, Embedded and Computing Technologies, 1(1), 21–25. https://doi.org/10.31838/JIVCT/01.01.05
4. Roukhami, M., Lazarescu, M. T., Gregoretti, F., Lahbib, Y., & Mami, A. (2019). Very low power neural network FPGA accelerators for tag-less remote person identification using capacitive sensors. IEEE Access, 7, 102217–102231. https://doi.org/10.1109/ACCESS.2019.2931392
5. Bharathkumar, K., Paolini, C., & Sarkar, M. (2020). FPGA-based edge inferencing for fall detection. In *2020 IEEE Global Humanitarian Technology Conference (GHTC)* (pp. 1–8). IEEE. https://doi.org/10.1109/GHTC46280.2020.9342948
6. Hasib-Al-Rashid, Manjunath, N. K., Paneliya, H., Hosseini, M., Hairston, W. D., & Mohsenin, T. (2020). A low-power LSTM processor for multi-channel brain EEG artifact detection. In *Proceedings of the 21st International Symposium on Quality Electronic Design (ISQED)* (pp. 105–110). IEEE. https://doi.org/10.1109/ISQED48828.2020.9137056
7. Chen, J., Hong, S., He, W., Moon, J., & Jun, S.-W. (2021). Eciton: Very low-power LSTM neural network accelerator for predictive maintenance at the edge. In *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)* (pp. 1–8). IEEE. https://doi.org/10.1109/FPL53798.2021.00009
8. Chen, J., Jun, S. W., Hong, S., He, W., & Moon, J. (2024). Eciton: Very low-power recurrent neural network accelerator for real-time inference at the edge. ACM Transactions on Reconfigurable Technology and Systems, 17. https://doi.org/10.1145/3629979
9. Bartels, J., Tokgoz, K. K., A. S., Fukawa, M., Otsubo, S., Li, C., et al. (2022). TinyCowNet: Memory- and power-minimized RNNs implementable on tiny edge devices for lifelong cow behavior distribution estimation. *IEEE Access, 10*, 32706–32727. https://doi.org/10.1109/ACCESS.2022.3156278
10. Al-Saud, F., & Al-Farsi, M. (2025). Energy efficient VLSI design for next generation IoT devices. Journal of Integrated VLSI, Embedded and Computing Technologies, 2(1), 46–52. https://doi.org/10.31838/JIVCT/02.01.06
11. Kreß, F., Serdyuk, A., Kobsar, D., Hotfilter, T., Höfer, J., Harbaum, T., & Becker, J. (2024) (2024). LOTTA: An FPGA-based low-power temporal convolutional network hardware accelerator. In *2024 IEEE 37th International System-on-Chip Conference (SOCC)* (pp. 126–131). IEEE. https://doi.org/10.1109/SOCC62300.2024.10737863
12. Qian, C., Ling, T., & Schiele, G. (2023). Enhancing energy-efficiency by solving the throughput bottleneck of LSTM cells for embedded FPGAs. In I. Koprinska et al. (Eds.), *Machine Learning and Principles and Practice of Knowledge Discovery in Databases* (pp. 594–605). Springer.
13. Szadkowski, Z., & Szadkowska, A. (2019). FPGA implementation of the trigger based on a fuzzy logic for a detection of cosmic rays in water Cherenkov detectors. In *2019 14th Conference on Industrial and Information Systems (ICIIS)* (pp. 431–435). IEEE. https://doi.org/10.1109/ICIIS47346.2019.9063266
14. Muralidharan, J. (2024). Optimization techniques for energy-efficient RF power amplifiers in wireless communication systems. SCCTS Journal of Embedded Systems Design and Applications, 1(1), 1–6. https://doi.org/10.31838/ESA/01.01.01
15. Matsui, T., Yukawa, C., Nagai, Y., Toyoshima, K., Hirata, A., & Oda, T. (2022). FPGA implementation of an interval type-2 fuzzy inference based wildfire monitoring system. In *2022 IEEE 11th Global Conference on Consumer Electronics (GCCE)* (pp. 876–877). IEEE. https://doi.org/10.1109/GCCE56475.2022.10014188
16. Valdez, R., Maldonado, Y., & Quevedo, J. (2023). Fuzzy hardware tool: An adaptable tool to facilitate the implementation of fuzzy inference systems in hardware. Electronics, 12, 2853. https://doi.org/10.3390/electronics12132853
17. Gao, L., Jiang, J., Xu, J., Wang, W., & Wu, P. (2025). SAPFIS: A parallel fuzzy inference system for air combat situation assessment. The Journal of Supercomputing, 81. https://doi.org/10.1007/s11227-024-06521-y
18. Gao, L., Jiang, J., & Xu, J. (2024). HPFIA: A high-performance fuzzy inference accelerator for situation assessment on airborne equipment. In *2024 IEEE International Conference on High Performance Computing and Communications (HPCC)* (pp. 456–465). IEEE. https://doi.org/10.1109/HPCC64274.2024.00068
19. Weste, N. H. E., & Harris, D. (2010). *CMOS VLSI design: A circuits and systems perspective* (4th ed.). Addison-Wesley.
20. Kang, S., Moon, J., & Jun, S. W. (2020). FPGA-accelerated time series mining on low-power IoT devices. In *2020 IEEE 31st International Conference on Application-Specific Systems, Architectures and Processors (ASAP)* (pp. 33–36). IEEE. https://doi.org/10.1109/ASAP49362.2020.00015
21. Singh, V., Yadav, A., & Gupta, S. (2023). Open circuit fault diagnosis and fault classification in multi-level inverter using fuzzy inference system. Serbian Journal of Electrical Engineering, 20, 163–189. https://doi.org/10.2298/SJEE2302163S
22. Bartels, J., Hagihara, A., Minati, L., Tokgoz, K. K., & Ito, H. (2023). An integer-only resource-minimized RNN on FPGA for low-frequency sensors in edge-AI. IEEE Sensors Journal, *23*, 17784–17793. https://doi.org/10.1109/JSEN.2023.3286580
23. Al-Yateem, N., Ismail, L., & Ahmad, M. (2024). A comprehensive analysis on semiconductor devices and circuits. Progress in Electronics and Communication Engineering, 2(1), 1–15. https://doi.org/10.31838/PECE/02.01.01