🧠 Neuromorphic RPU for Robotics – Design & Verification Overview

Designing and verifying a **Robotic Processing Unit (RPU)** based on **neuromorphic principles** is a highly innovative and research-intensive project. This architecture is inspired by the human brain's structure and behavior and aims to enhance robotic intelligence with real-time, energy-efficient processing. Here's a structured overview of such a project from both **design** and **verification** perspectives:

🎯 Objective

To design and verify a **neuromorphic Robotic Processing Unit (RPU)** that mimics biological neural networks for real-time, low-power decision-making in robotics applications (e.g., autonomous navigation, object manipulation, learning from environment).

🧩 1. Architecture Overview

🔹 Core Components:

Block	Description
Neuron Core	Digital spiking neuron model (e.g., LIF – Leaky Integrate & Fire)
Synapse Array	Memory-mapped or crossbar-based STDP-enabled synapse model
Axon Interface	Manages spike routing and communication between neurons
RPU Controller	FSM/Processor to manage learning modes, spike timing, and configuration
Sensor Interface	Integrates with external sensors (camera, LIDAR, pressure, etc.)
Actuator Driver	Sends commands to motors based on spike-based decision
Learning Engine	Implements Hebbian or STDP (Spike-Timing Dependent Plasticity) learning rules
NoC / Bus	Interconnect for scalable neuron-synapse network

🔧 2. Design Specifications

HDL/RTL:

- Language: Verilog / SystemVerilog
- Clock domains: Low-frequency main clock + asynchronous event-based clocks
- Memory: Dual-port SRAM / CAM for synapse weights
- Optional: Implement spike compression for low-bandwidth transmission

Design Features:

- Event-driven processing
- Local learning (online STDP)
- Configurable neuron and synapse parameters
- GPIO/I2C/SPI for sensor integration

✅ 3. Verification Plan

Testbench Architecture:

- **UVM-based Testbench**
  - **Agent**: For sensor/actuator interfaces
  - **Sequencer**: Stimuli for spike inputs
  - **Driver**: Converts stimuli to signals
  - **Monitor**: Captures outputs and spike events
  - **Scoreboard**: Compares actual vs expected spike train / behavior
  - **Coverage**: Functional + Code coverage (SystemVerilog + UCIS)

Stimulus:

- Spiking patterns (Poisson, bursty, periodic)
- Sensor input models (vision, pressure, IMU)
- Timing variations (jitter, delayed response)
- Learning behavior validation (weight updates, neuron plasticity)

📊 4. Functional Verification Metrics

- Neuron firing thresholds, refractory periods
- Spike propagation and timing
- STDP weight adaptation over time
- Sensor-to-action latency
- Actuator command validity under stimulus

📈 5. Optional Extensions

- Use **THEJAS32 / RISC-V core** as control processor
- Neuromorphic co-processor in **Edge AI** robot
- Connect RPU to **ESP32/ESP8266** for cloud telemetry
- Integrate with **Loihi/Nirvana-inspired chiplets**

🧪 6. Simulation & Tools

- **EDA Tools**: Synopsys VCS / Cadence Xcelium / QuestaSim
- **Waveform Viewer**: GTKWave or DVE
- **RTL Synthesis**: Yosys (open) or Design Compiler
- **FPGA Prototyping**: Artix-7, Cyclone V, or Arty-A7
- **Formal Verification**: JasperGold / OneSpin

📚 7. Documentation for VLSID Conferences

Include in your paper:

- Block diagrams of neuron/synapse/spike path
- RTL microarchitecture
- Simulation results (e.g., spike response to sensor)
- FPGA prototype results (if available)
- Energy comparison with traditional CPU/GPU-based control
- Use-case demo: obstacle-avoiding robot, robotic arm learning grasp

Would you like:

- 🧠 Synthesizable RTL for neuron/synapse blocks and connection protocols ?
- 📄 VLSID/IEEE-style 6-page research paper template?
- 🧪 UVM testbench code structure?