Key Components for SNN Design on Hardware

MAJOR PROJECT REPORT

Submitted to:

Department of Electronics and Communication Engineering

Submitted by:

Name — Vedika Jain
Roll Number — 21BEC018

Under the Supervision of:
Prof. Sajad Ahmad Loan

Faculty of Engineering and Technology

Jamia Millia Islamia, New Delhi

Certificate

This is to certify that the project report entitled 'Key Components for SNN Design on Hardware' submitted by Vedika Jain (21BEC018) to the Department of Electronics and Communication Engineering, Faculty of Engineering and Technology, Jamia Millia Islamia, New Delhi– 110025 in partial fulfilment for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering is a bonafide record of project work carried out by her under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institution or University for the award of any degree.

Prof. Sajad Ahmad Loan
Supervisor
D/o E&C Engg.
FET, JMI

Countersignature of HoD with seal

New Delhi

16 May 2025

Declaration

I hereby declare that the major project report entitled 'Key Components for SNN Design on Hardware' submitted to the Department of Electronics and Communication Engineering, Faculty of Engineering and Technology, Jamia Millia Islamia, New Delhi– 110025 in partial fulfilment for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering is a bonafide record of project work carried out by me. The contents of this report, in full or in parts, have not been submitted to any other Institution or University for the award of any degree or diploma.

Vedika Jain

21BEC018

New Delhi

16 May 2025

List of Contents

Acknowledgements

It’s by the Almighty’s grace that this world goes round. I bow my head before the Lord for helping me work on this project. I will forever be indebted to my family, my parents and my sister, for looking out for me every step along the way. Without their unwavering support, this project report wouldn’t have come to existence.

I extend my gratitude to my supervisor, Prof. Sajad Ahmad Loan, for his patience and guidance throughout the project. The report stands in its present form thanks to his probing counter questions and keen insights.

Introduction

Traditional computing faces challenges in tasks such as pattern recognition, sensory processing, and real-time interaction, which are effortlessly performed by the human brain. AI is best run on hardware which is similar to the abstraction. Neuromorphic computing, inspired by the brain's efficiency and parallelism, aims to overcome these limitations. Neuromorphic computing seeks to mimic the structure and function of the human brain using electronic circuits. Present systems make use of the second generation ANNs.

Third-generation ANNs are Spiking Neural Networks (SNNs) that use biologically plausible spiking neurons as the basic computational units. Neural information in the spiking neuron is transmitted and processed by precisely timed spike trains. With SNNs, there exists a trade-off between accuracy and simplicity. Compared with the first- and second-generation ANN models, SNNs can describe the real biological nervous system more accurately, so as to achieve efficient information processing. These networks are well-suited for FPGA implementation due to their parallel nature.

This project focuses on arriving at informed design decisions for realising an SNN for temporal processing applications. The minor project looked at the available choices for the neural network components. The major project looks at available benchmarks and FPGA platforms. It also works towards efficient hardware implementation of crucial modules for designing the SNN. The Izhikevich neuron is based on the CORDIC algorithm, and STDP is based on the Base-2 approximation, both of which reduce multiplier use and rely on cheap addition and shift operations.

Spiking Heidelberg Digits Dataset

The foremost requirement for designing a neural network is a dataset to train and test the design. Most datasets available have been compiled for ANNs. There are no unified approaches to measuring performance in SNNs which is partially due to the numerous different learning approaches and architectures. Spiking Heidelberg Digits (SHD) dataset is a step in the direction towards providing a benchmark to test the spatiotemporal processing and pattern recognition in SNNs.

The SHD dataset is a spiking version of the Heidelberg Digits dataset. It is an audio-based classification dataset of 1k spoken digits ranging from zero to nine in the English and German languages. The audio waveforms have been converted into spike trains using an artificial model of the inner ear and parts of the ascending auditory pathway. The SHD dataset has 8,156 training and 2,264 test samples.

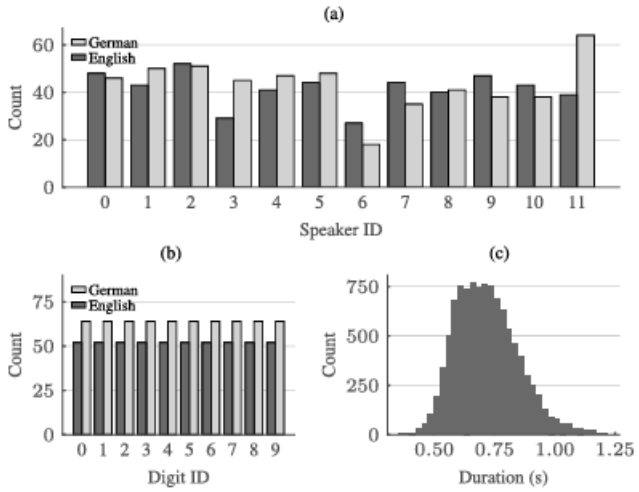


Fig. 1. The Heidelberg Digits HD have a balanced class count and variable temporal duration. The HD consist of 10420 recordings of spoken digits ranging from *zero* to *nine* in English and German language. (a) Histogram of per-speaker digit counts. Variable numbers of digits are available for each speaker and each language. (b) Histogram of per-class digit counts. The data set is balanced in terms of digits within each language. (c) Histogram of audio recording durations. The HD audio recordings were cut for minimal duration to keep computation time at bay.

Statistical Overview of the original Heidelberg Digits (HD) dataset

This dataset removes the need to dedicate resources for spike encoding and decoding. It can be augmented to replicate real-life scenarios by adding noise and other distortions, given its high fidelity.

CORDIC Based Izhikevich Neuron

COordinate Rotation DIgital Computer (CORDIC) is an iterative algorithm generalized for calculation of hyperbolic and exponential functions, multiplications, divisions and square roots. CORDIC only requires simple shift and addition operations, which can be cheaply implemented on hardware hence making it an appropriate choice for fast and low-cost hardware implementations.

Verilog Module

```
// CORDIC-based Izhikevich Neuron Implementation in Verilog

module izhikevich_neuron #(parameter DT = 16'h0100) // dt = 1.0 in Q8.8
(
    input wire clk,
    input wire rst,
    input wire [15:0] I,          // Input current (Q8.8)
    output reg spike              // Spike flag
);

    // Internal state variables (Q8.8)
    reg signed [15:0] v, u;

    // Parameters for regular spiking neuron
    wire signed [15:0] a = 16'h0051;   // a = 0.02
    wire signed [15:0] b = 16'h0200;   // b = 0.2
    wire signed [15:0] c = -16'sd16640; // c = -65 -> -65 * 256
    wire signed [15:0] d = 16'h0800;   // d = 8

    wire signed [15:0] v_sq;
    wire signed [15:0] term1, term2, term3, dv_raw, du_raw;
    wire signed [15:0] bv;

    // Threshold
    localparam signed [15:0] V_THRESH = 16'd7680; // 30 * 256

    // Squaring v using multiplier (CORDIC-based or simplified)
    cordic_mult sq_v (.x(v), .y(v), .out(v_sq));

    // term1 = 0.04 * v^2
    cordic_mult mul_004 (.x(16'h0148), .y(v_sq), .out(term1));
    // term2 = 5 * v
    cordic_mult mul_5v (.x(16'h0500), .y(v), .out(term2));
    // term3 = 140
    assign term3 = 16'd35840; // 140 * 256
```

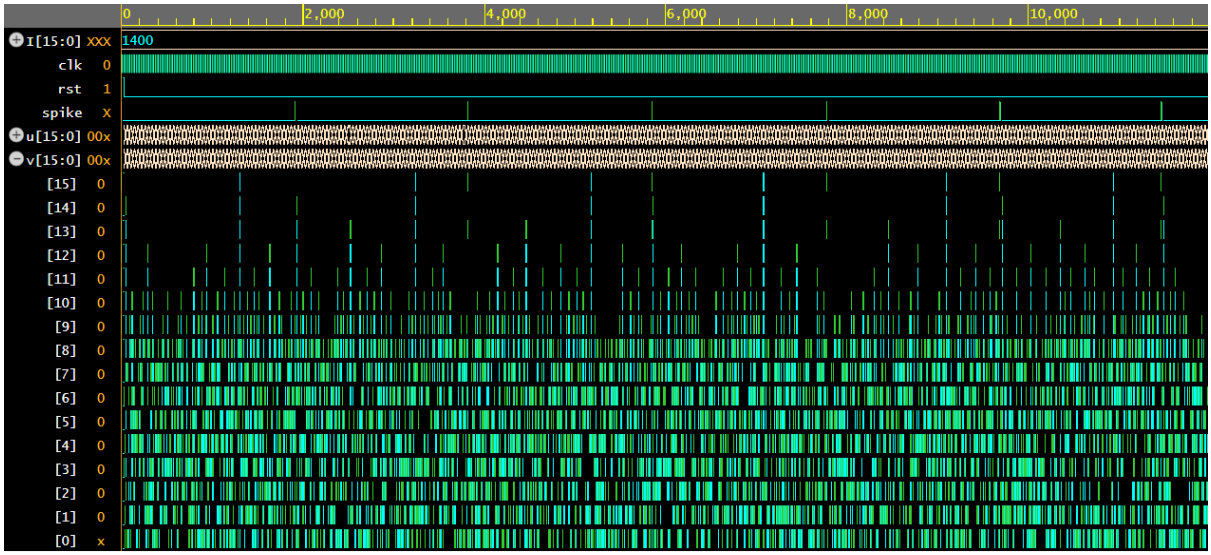
```
    assign dv_raw = term1 + term2 + term3 - u + I;

    cordic_mult mul_bv (.x(b), .y(v), .out(bv));
    assign du_raw = (bv - u);

    wire signed [15:0] du;
    cordic_mult mul_a (.x(a), .y(du_raw), .out(du));

    // Euler integration step
    always @(posedge clk or posedge rst) begin
        if (rst) begin
            v <= c;
            u <= 0;
            spike <= 0;
        end else begin
            if (v >= V_THRESH) begin
                v <= c;
                u <= u + d;
                spike <= 1;
            end else begin
                v <= v + ((dv_raw * DT) >>> 8); // Q8.8 multiply with dt, shift to maintain scale
                u <= u + ((du * DT) >>> 8);
                spike <= 0;
            end
        end
    end

endmodule

// Simplified CORDIC Multiplier Module (Shift-Add for Q8.8)
module cordic_mult(
    input signed [15:0] x,
    input signed [15:0] y,
    output signed [15:0] out
);
    wire signed [31:0] temp;
    assign temp = x * y;
    assign out = temp[23:8]; // Q8.8 result
endmodule
```

Testbench

```
module izhikevich_neuron_tb;
    reg clk = 0;
```

```
    reg rst = 1;
    reg signed [15:0] I;
    wire spike;

    izhikevich_neuron uut (
        .clk(clk),
        .rst(rst),
        .I(I),
        .spike(spike)
    );

    always #5 clk = ~clk; // 100MHz clock

    initial begin
        $dumpfile("izhikevich_neuron.vcd");
        $dumpvars(0, izhikevich_neuron_tb);

        #20 rst = 0;
        I = 16'd5120; // ~20.0 in Q8.8

        repeat (2000) @(posedge clk);

        $finish;
    end
endmodule
```

Result



Base-2 Approximated STDP

STDP equations are made from exponential functions. Accuracy and hardware cost are the main concerns simultaneously when a large-scale implementation is targeted. The exponential function can be converted to a base-2 function.

$$exp(x) \cong 2^{x+2^{-1}x-2^{-4}x} \cong 2^{1.4375x}$$

The approximated equations therefore become:

$$\Delta w = \begin{cases} \Delta w^+ = A^+ .2^{-1.4375(\frac{\Delta t}{\tau^+})}, & \text{if } \Delta t \geq 0 \\ \Delta w^- = -A^- .2^{1.4375(\frac{\Delta t}{\tau^-})}, & \text{if } \Delta t \leq 0 \end{cases}$$

Verilog Module

```
module base2_pstdp (
    input wire clk,
    input wire rst,
    input wire pre_spike,
    input wire post_spike,
    output reg [15:0] weight
);
    parameter A_plus  = 16'd32;    // LTP step
    parameter A_minus = 16'd32;    // LTD step
    parameter TAU_SHIFT = 2;       // Shift = 2 -> divide delta_t by 4

    reg [15:0] pre_time = 0;
    reg [15:0] post_time = 0;
    reg [15:0] delta_t;
    reg [15:0] delta_w;

    always @(posedge clk or posedge rst) begin
        if (rst) begin
            pre_time  <= 0;
            post_time <= 0;
            weight    <= 16'd100; // Initial weight
        end else begin
            // Time counters increment every clock
            pre_time  <= pre_time + 1;
            post_time <= post_time + 1;

            if (pre_spike)
                pre_time <= 0;
```

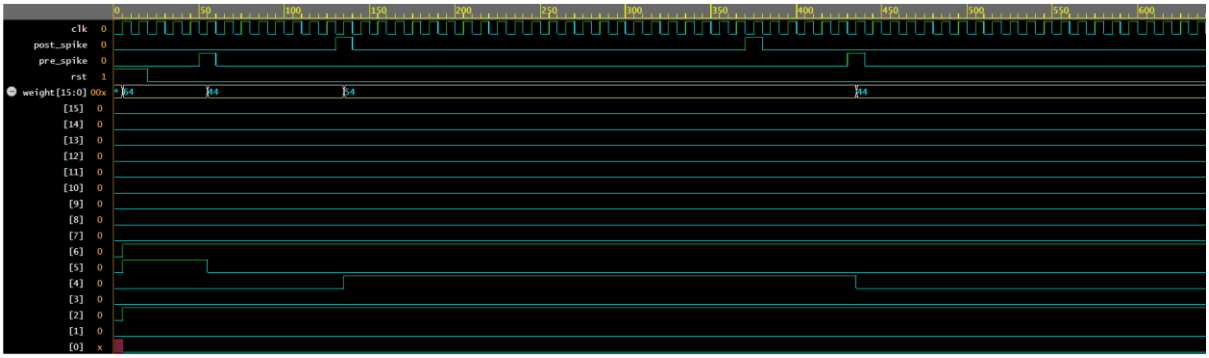
```
        if (post_spike)
            post_time <= 0;

        if (pre_spike && post_time < 16'd255) begin
            // Post followed by pre → LTD
            delta_t = post_time;
            delta_w = A_minus >> (delta_t >> TAU_SHIFT);
            weight <= (weight > delta_w) ? weight - delta_w : 0;
        end else if (post_spike && pre_time < 16'd255) begin
            // Pre followed by post → LTP
            delta_t = pre_time;
            delta_w = A_plus >> (delta_t >> TAU_SHIFT);
            weight <= (weight + delta_w < 16'hFFFF) ? weight + delta_w : 16'hFFFF;
        end
    end
  end
endmodule
```

Testbench

```
// Testbench
module base2_pstdp_tb;
  reg clk = 0;
  reg rst = 1;
  reg pre_spike = 0;
  reg post_spike = 0;
  wire [15:0] weight;

  base2_pstdp uut (
    .clk(clk),
    .rst(rst),
    .pre_spike(pre_spike),
    .post_spike(post_spike),
    .weight(weight)
  );

  always #5 clk = ~clk; // 100MHz clock

  initial begin
    $dumpfile("base2_pstdp.vcd");
    $dumpvars(0, base2_pstdp_tb);

    // Reset pulse
    #20 rst = 0;
```

```
        // Test LTP: Pre then Post
        #30 pre_spike = 1; #10 pre_spike = 0;
        #70 post_spike = 1; #10 post_spike = 0;

        // Wait
        #200;

        // Test LTD: Post then Pre
        #30 post_spike = 1; #10 post_spike = 0;
        #50 pre_spike = 1; #10 pre_spike = 0;

        // Wait more
        #200;

        $finish;
    end
endmodule
```

Result



Xilinx Artix-7 (XC7A35T) FPGA

For hardware realization of the neuromorphic spiking neural network, this project targets a Field-Programmable Gate Array (FPGA) as the implementation platform. FPGAs offer massive parallelism, low-latency computation, and reconfigurability, making them well-suited for event-driven models such as spiking neural networks (SNNs).

This project uses the Xilinx Artix-7 series FPGA as a representative platform, due to its balance of logic resources, memory, clocking infrastructure, and cost-effectiveness. The Artix-7 family is supported by the Vivado Design Suite, which enables synthesis, simulation, floorplanning, and timing analysis.

Key Advantages of Using an FPGA for SNNs:

- **Parallelism**: Multiple neuron and synapse modules can operate concurrently, reflecting the nature of biological networks.
- **Timing Precision**: Spikes can be processed at clock-level resolution, enabling fine-grained control over temporal dynamics.
- **Low Latency**: Deterministic hardware execution paths allow real-time processing of spike events.
- **Reconfigurability**: The neuron model, learning rule (e.g., STDP), or network topology can be updated post-synthesis by reprogramming the bitstream.
- **Hardware-Efficient Arithmetic**: Fixed-point arithmetic (e.g., Q8.8) can be used to replace floating-point computations, significantly reducing area and power.

Device Overview: Xilinx Artix-7 XC7A35T

The XC7A35T is a low-to-mid capacity FPGA suitable for academic, research, and low-power edge deployments.

Specification	Value
Logic Cells	33,280
LUTs (Look-Up Tables)	20,800
Flip-Flops	41,600
Block RAM	50 × 36 Kb
DSP Slices	90
I/O Pins	210
Tool Support	Vivado Design Suite
Packaging	CPG236 (for Basys 3 board)

This platform is sufficient to prototype small- to medium-scale spiking networks, and can be scaled up to higher-capacity FPGAs (e.g., Kintex or Virtex series) for more complex models or denser connectivity.

A development board based on the XC7A35T

Conclusion

The project culminates to serve as a guide towards a biologically plausible, efficient, and accurate spiking neural network on hardware, the platform of interest being field programmable gate arrays.

It looked at the SHD dataset which offers a rich collection of high-quality sample audios, in a convenient form. It can be readily used to train, test, and benchmark networks, removing the need to dedicate resources for spike encoding and decoding. It can be augmented to replicate real-life scenarios by adding noise and other distortions.

How well a neural network in theory performs practically depends on its implementation. Thus, we look at CORDIC Izhikevich neuron and Base-2 STDP which simplify multiplications and exponentiations as shifts and additions.

Finally, we look at a low power FPGA platform which can support academic research.

Scope for Improvement

The graduation project had originally intended to implement an SNN on an FPGA, but the task turned out to be non-trivial. SNN performance highly depends on implementation and research in this field is not mature yet. Multiple iterations and training resources are needed to arrive at a network which is both efficient and accurate. Thus, the project instead endeavoured to look at the latest work in the domain and zero down on the best available options available to someone who may venture in this direction.

Bibliography

[1] B. Cramer, Y. Stradmann, J. Schemmel and F. Zenke, "The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks," in IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 7, pp. 2744-2757, July 2022, doi: 10.1109/TNNLS.2020.3044364.

[2] E. M. Izhikevich, "Simple model of spiking neurons," in IEEE Transactions on Neural Networks, vol. 14, no. 6, pp. 1569-1572, Nov. 2003, doi: 10.1109/TNN.2003.820440.

[3] M. Heidarpur, A. Ahmadi, M. Ahmadi and M. Rahimi Azghadi, "CORDIC-SNN: On-FPGA STDP Learning With Izhikevich Neurons," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 7, pp. 2651-2661, July 2019, doi: 10.1109/TCSI.2019.2899356.

[4] S. Gomar and M. Ahmadi, "Digital Realization of PSTDP and TSTDP Learning," 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 2018, pp. 1-5, doi: 10.1109/IJCNN.2018.8489263.

[5] Ricklin, Steffen, et al. "Neuron Model Complexity." (2023).

[6] Yu, Chengting, et al. "Stsc-snn: Spatio-temporal synaptic connection with temporal convolution and attention for spiking neural networks." *Frontiers in Neuroscience* 16 (2022): 1079357.

[7] T. Fernandez-Hart, T. Kalganova and J. C. Knight, "Exploring 8-Bit Arithmetic for Training Spiking Neural Networks," 2024 IEEE International Conference on Omni-layer Intelligent Systems (COINS), London, United Kingdom, 2024, pp. 1-6, doi: 10.1109/COINS61597.2024.10622154.