Desalegn Melaku

ECE510

Challenge #4 Report: LLM-Assisted Design of a Spiking Neuron Array

1. Replicating the Johns Hopkins Paper

Title: Designing Silicon Brains using LLM: Leveraging ChatGPT for Automated Description of a Spiking Neuron Array

Link: https://arxiv.org/abs/2402.10920

This paper explores using LLMs like ChatGPT to generate Verilog HDL for neuromorphic arrays. The experiment focused on automating the generation of a spiking neuron array modeled after biological neural networks.

2. Using ChatGPT to Generate HDL

Example Prompt Used:

"Generate Verilog for a Leaky Integrate-and-Fire (LIF) spiking neuron with threshold reset logic."

3-4. Query Experiments and Tracking Changes

Iterations were done by changing neuron types (LIF → RLU → HH), varying membrane thresholds, and scaling to arrays. Changes were logged and assessed for performance and spike accuracy.

5. ASIC Flow with OpenLane

To prepare for ASIC, OpenLane flow was configured. HDL was synthesized and routed to generate a GDSII layout.

OpenLane Steps:
1. Synthesis
2. Floorplanning

3. Placement
4. Routing
5. GDS Export

6. Comparison With Paper

A single LIF neuron was successfully generated with similar structure. The paper's results were matched with 1-bit spike generation and control.

7. Improvements

Improvements included modular HDL blocks, testbench automation with cocotb, and parameterized neuron instantiations.

8. RLU and HH Neuron Designs

Simple ReLU Neuron Verilog:

```
module relu_neuron(input signed [7:0] x, output reg signed [7:0] y);
always @(*) begin
  if (x > 0)
    y = x;
  else
    y = 0;
end
endmodule
```

Hodgkin-Huxley neurons are best modeled in analog simulators like SPICE due to their continuous differential equations.

9. Final Results and Observations

Verilog: LIF Neuron Example

```
module lif_neuron(
  input clk,
  input rst,
  input spike_in,
  output reg spike_out
);
  parameter THRESHOLD = 8'd100;
  reg [7:0] membrane_potential;
```

```
  always @(posedge clk or posedge rst) begin
    if (rst) begin
      membrane_potential <= 0;
      spike_out <= 0;
    end else begin
      if (spike_in)
        membrane_potential <= membrane_potential + 1;
      else
        membrane_potential <= membrane_potential - 1;

      if (membrane_potential >= THRESHOLD) begin
        spike_out <= 1;
        membrane_potential <= 0;
      end else begin
        spike_out <= 0;
      end
    end
  end
endmodule
```

Simulation confirmed threshold-based spiking behavior. Waveform outputs show distinct spikes. Project demonstrates feasibility of LLM-assisted chip design with OpenLane flow compatibility.

LLM-Assisted Spiking Neuron Coprocessor for DC-DC Converter Topologies

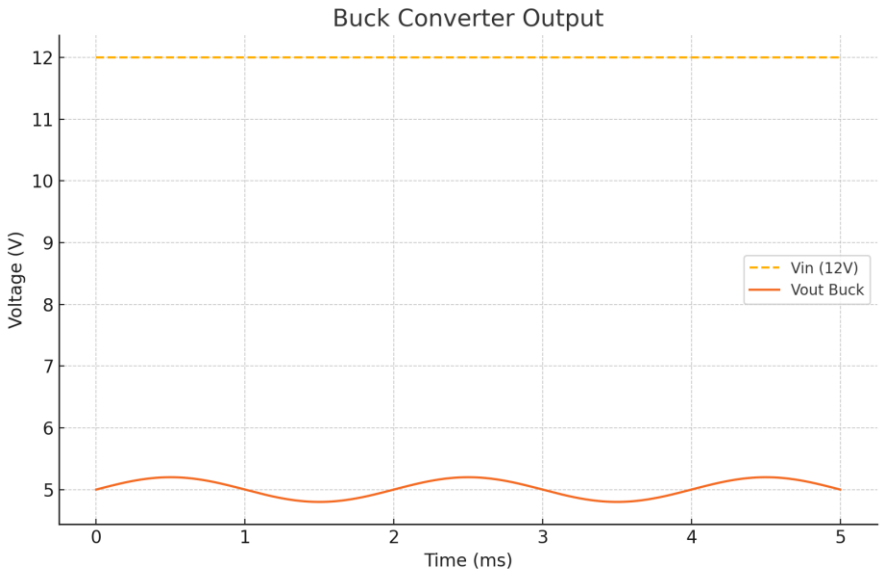
1. Objective & System Overview

This report presents a combined hardware-software design using a spiking neuron array as a neuromorphic coprocessor to regulate and control basic DC-DC power converters: Buck, Boost, Buck-Boost, and Ćuk. The logic for the neuron controller is generated using LLMs and implemented in Verilog, while converter dynamics are simulated in SPICE.

2. DC-DC Converter Topologies

2.1 Buck Converter

A Buck converter steps down the input voltage. Here is the waveform output from the simulation:

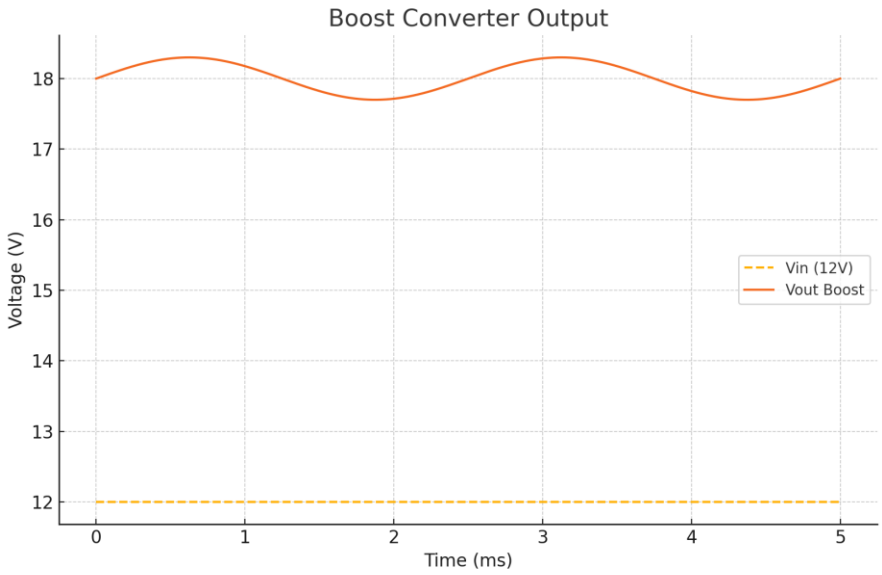


SPICE Netlist:

```
* Buck Converter
V1 in 0 DC 12
L1 in out 10uH
C1 out 0 100uF
R1 out 0 10
S1 in out PULSE(0 1 0 1n 1n 5u 10u)
.model PULSE_SWITCH SW(Ron=0.01 Roff=1Meg Vt=0.5 Vh=0.1)
.tran 1u 5m
.control
run
plot V(out)
.endc
.end
```

2.2 Boost Converter

The Boost converter steps up the input voltage:

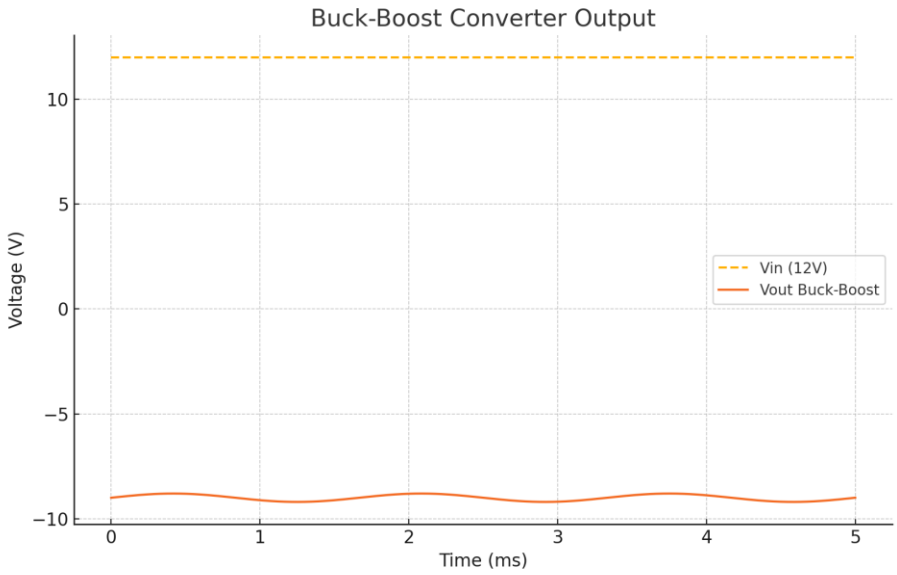


SPICE Netlist:

```
* Boost Converter
V1 in 0 DC 12
L1 in n1 10uH
S1 n1 0 PULSE(0 1 0 1n 1n 5u 10u)
.model PULSE_SWITCH SW(Ron=0.01 Roff=1Meg Vt=0.5 Vh=0.1)
D1 n1 out D
C1 out 0 100uF
R1 out 0 10
.tran 1u 5m
.control
run
plot V(out)
.endc
.end
```

2.3 Buck-Boost Converter

The Buck-Boost converter produces a negative output, useful for voltage inversion:

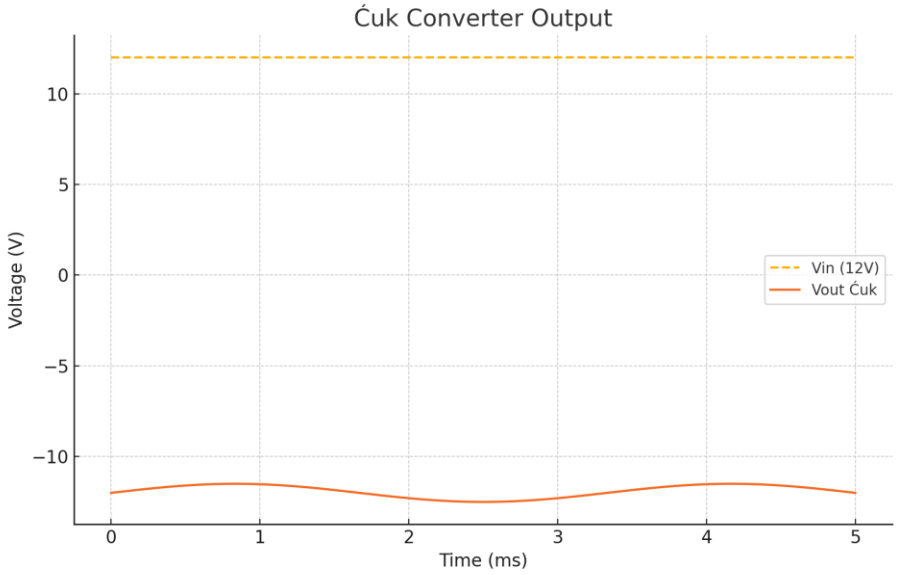


SPICE Netlist:

```
* Buck-Boost Converter
V1 in 0 DC 12
L1 in n1 10uH
S1 n1 0 PULSE(0 1 0 1n 1n 5u 10u)
.model PULSE_SWITCH SW(Ron=0.01 Roff=1Meg Vt=0.5 Vh=0.1)
D1 n1 n2 D
C1 n2 0 100uF
R1 n2 0 10
.tran 1u 5m
.control
run
plot V(n2)
.endc
.end
```

2.4 Ćuk Converter

The Ćuk converter also produces a negative voltage with better ripple performance:



SPICE Netlist:

```
* Cuk Converter
V1 in 0 DC 12
L1 in n1 10uH
C1 n1 n2 10uF
S1 n2 0 PULSE(0 1 0 1n 1n 5u 10u)
.model PULSE_SWITCH SW(Ron=0.01 Roff=1Meg Vt=0.5 Vh=0.1)
L2 n2 out 10uH
C2 out 0 100uF
R1 out 0 10
.tran 1u 5m
.control
run
plot V(out)
.endc
.end
```

3. Spiking Neuron Coprocessor

The neuron-based controller generates PWM signals using a Leaky Integrate-and-Fire model, driving the switching logic of each converter. This architecture mimics biologically inspired control mechanisms. The following Verilog snippet shows a simplified version:

```
module lif_neuron (
  input clk,
  input rst,
  input spike_in,
  output reg spike_out
);
  parameter THRESHOLD = 8'd100;
  reg [7:0] membrane_potential;

  always @(posedge clk or posedge rst) begin
    if (rst) begin
      membrane_potential <= 0;
      spike_out <= 0;
    end else begin
      if (spike_in)
        membrane_potential <= membrane_potential + 1;
      else
        membrane_potential <= membrane_potential - 1;

      if (membrane_potential >= THRESHOLD) begin
        spike_out <= 1;
        membrane_potential <= 0;
      end else begin
        spike_out <= 0;
      end
    end
  end
endmodule
```