SRAM Based Neural Network System for Traffic-Light Recognition in Autonomous Vehicles

Yuji Murakami, Yuichi Okuyama, Abderazek Ben Abdallah
Graduate School of Computer Science and Engineering,
Adaptive Systems Laboratory
The University of Aizu, Japan

Outline

- Background
- Motivation
- Paper Contribution
- System Architecture
- Preliminary Evaluation
- Conclusion and Future work

Outline

- Background
- Motivation
- Paper Contribution
- System Architecture
- Evaluation
- Conclusion and Future work

Background (1/5)

- To realize autonomous vehicles, image recognition with high accuracy and high speed is necessary in the vehicle environment.

- It is difficult to satisfy this constraint with software implementation.

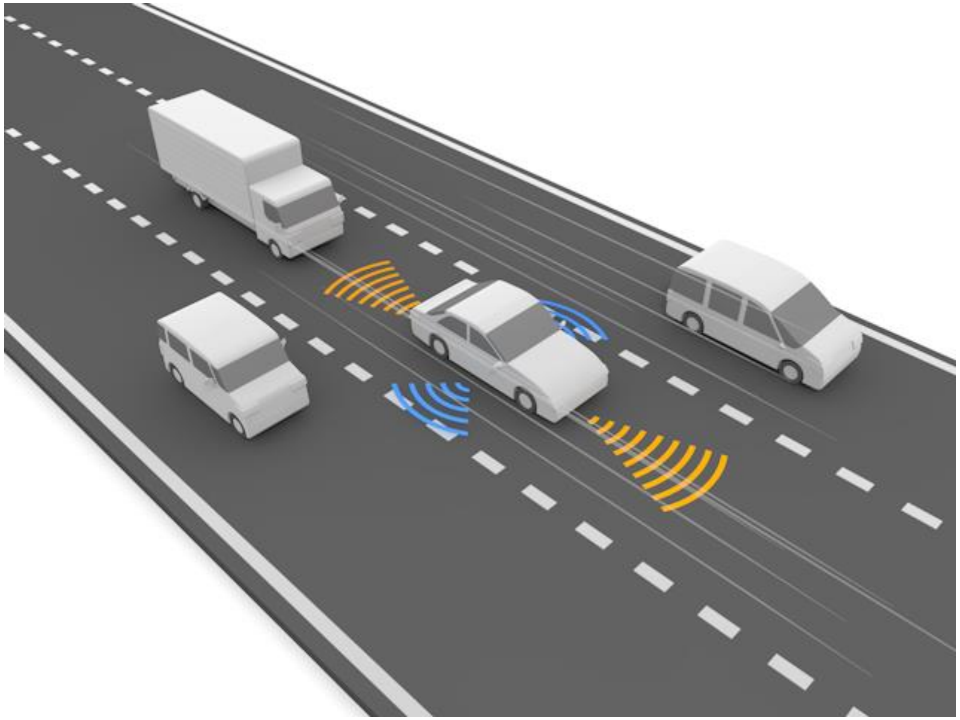


Figure 1. Autonomous vehicles Example

Background (2/5)

- DNNs are recently used in many machine learning applications, from speech recognition and natural language processing, to computer vision, and image recognition.

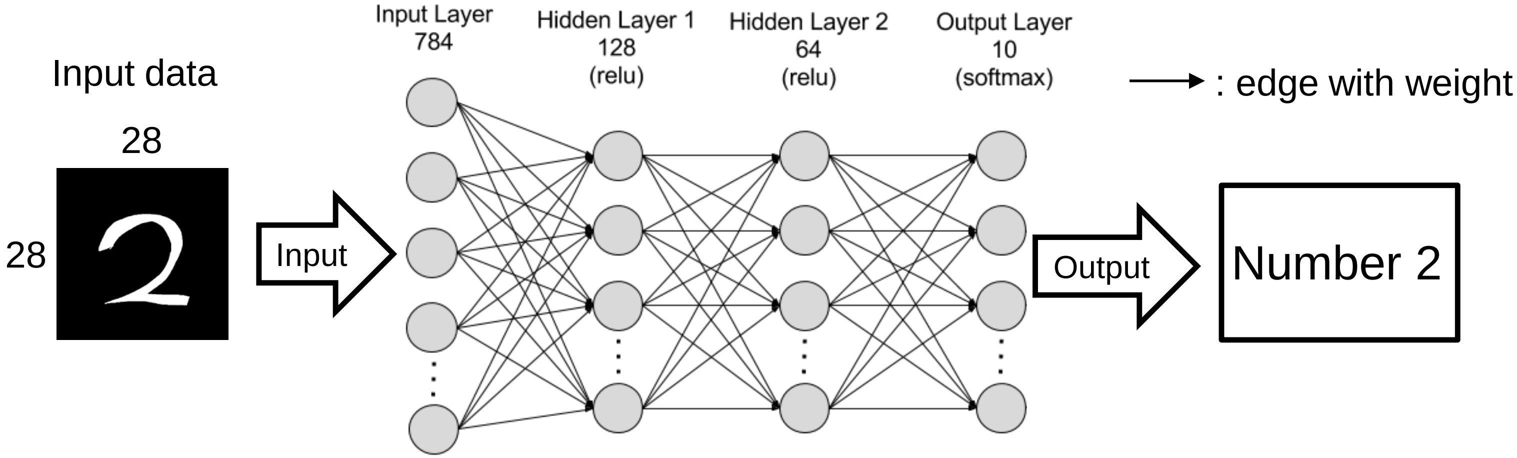


Figure 2. Example of handwritten character recognition with Deep Neural Network

- Hardware implementations of large DNN/ANNs offer superior execution speed compared to sequential SW approaches due to the inherent parallelism of HW.

Background: Inputs To Neurons (3/5)

- **Inputs x_i** arrive through pre-synaptic connections
- Synaptic efficacy is modeled using real **weights w_i**
- The activate of the neuron is a **nonlinear function *f*** of its weighted inputs

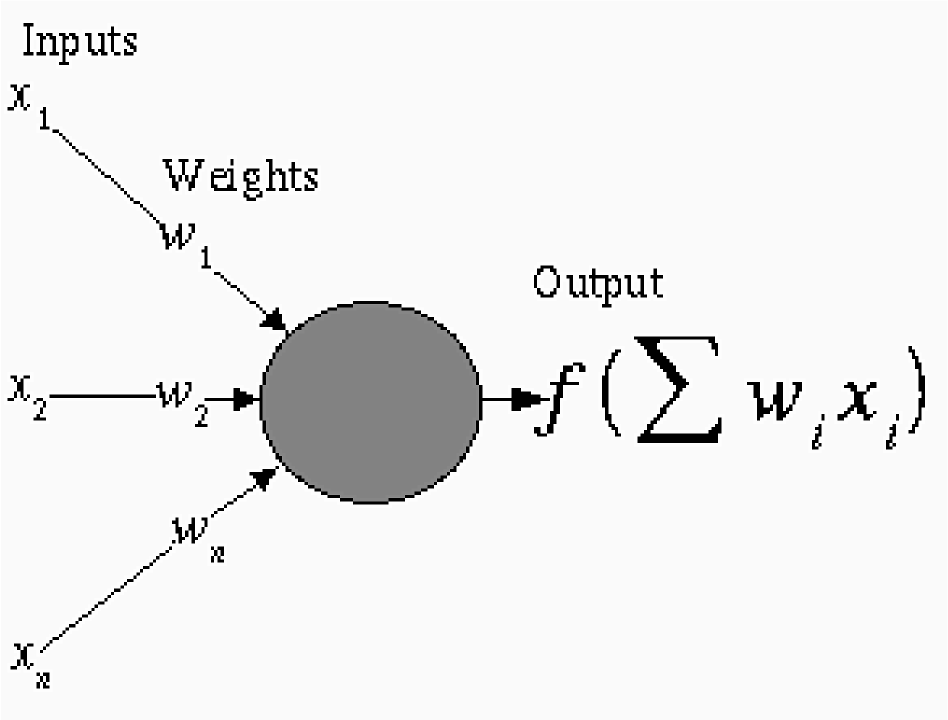


Fig. 3 Basic Neuron Model

Background:
Output from Neurons (4/5)

- The activate function is normally nonlinear
- Models include
  - Sigmoid

$$f(x) = \frac{1}{1+e^{-\lambda x}}$$

  - Piecewise linear

$$f(x) = \begin{cases} x, if \;\; x \geq \theta \\ 0, if \;\; x < \theta \end{cases}$$

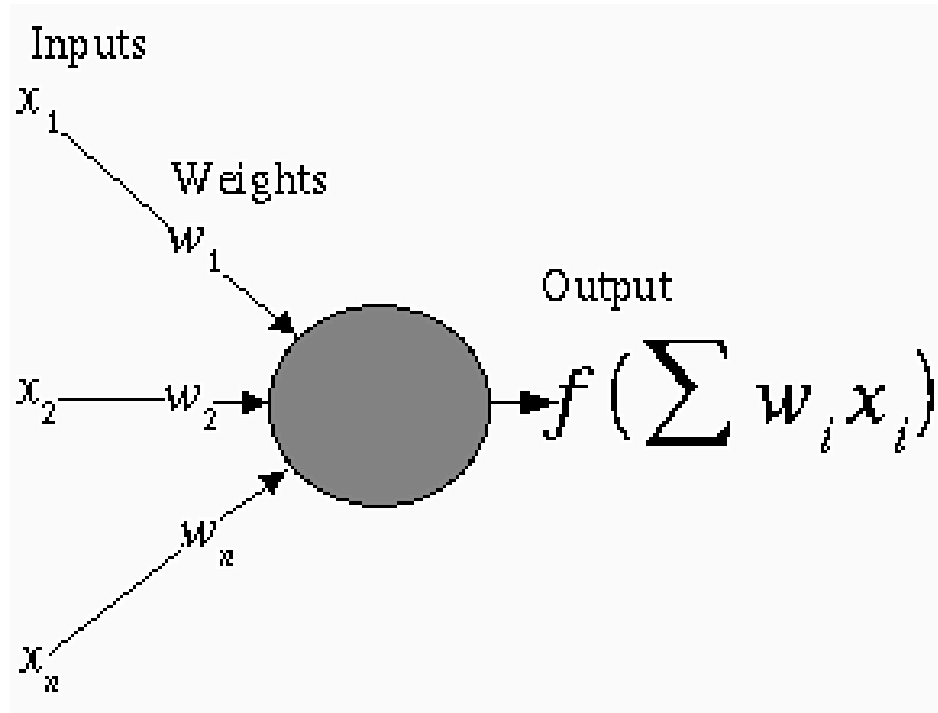


Fig. 3 Basic Neuron Model

Background (5/5)

- Large DNN models have proven to be very powerful, but implementing energy-efficient DNN in ASIC/FPGA is still a challenging task:
  - Because the required computations consume large amounts of energy
  - Large memory to store the weights
    - i.e. >100M parameters for large speech recognition tasks
  - Large wiring overhead exists due to a large number of connections between neurons.

Outline

- Background
- Motivation
- Paper Contribution
- System Architecture
- Preliminary Evaluation
- Conclusion and Future work

Motivation

- Miss-recognition of traffic-light in autonomous vehicles may lead to serious accidents and damages.

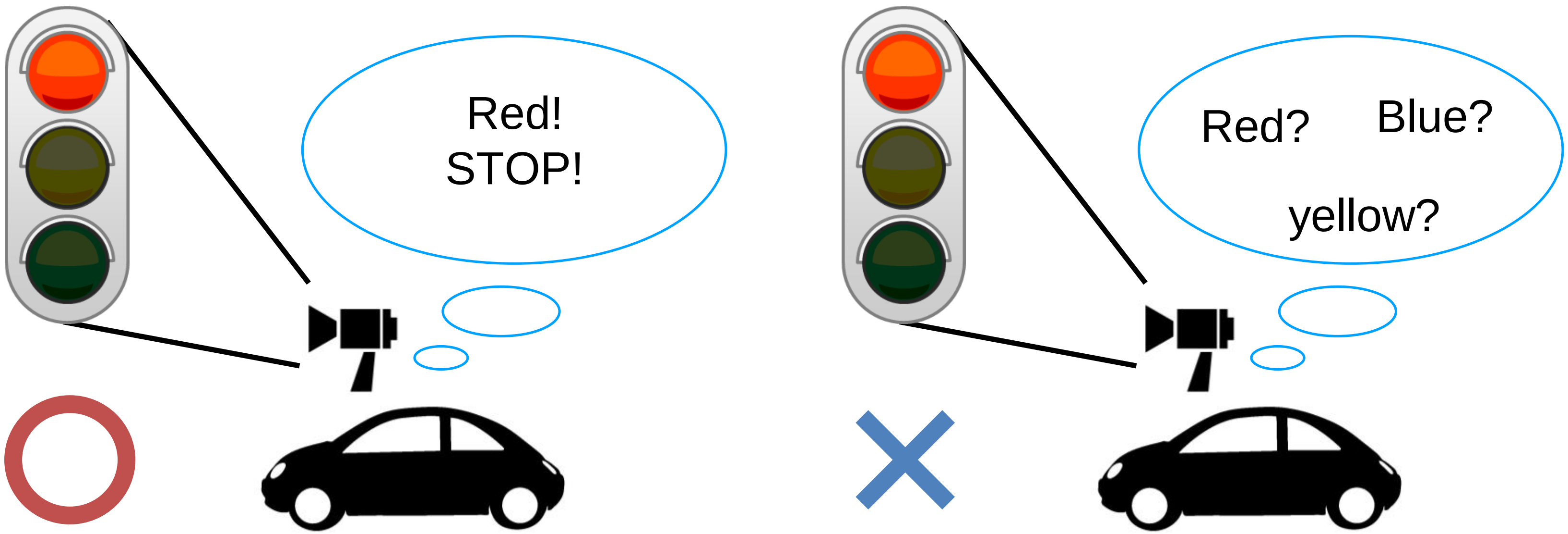


Figure 4: Traffic-Light Recognition in Autonomous Vehicles.

Motivation

- Spiking Neural Network (SNN) is an efficient approach with a much higher realism than earlier ANNs:
  - Large-scale HW accelerated neural simulations.
  - Real-time behaving.
  - Enables richer interactions with neuroscience.
- SNN applications:
  - Hardware IP cores, especially in embedded/ubiquitous systems, where power efficiency is a major focus.

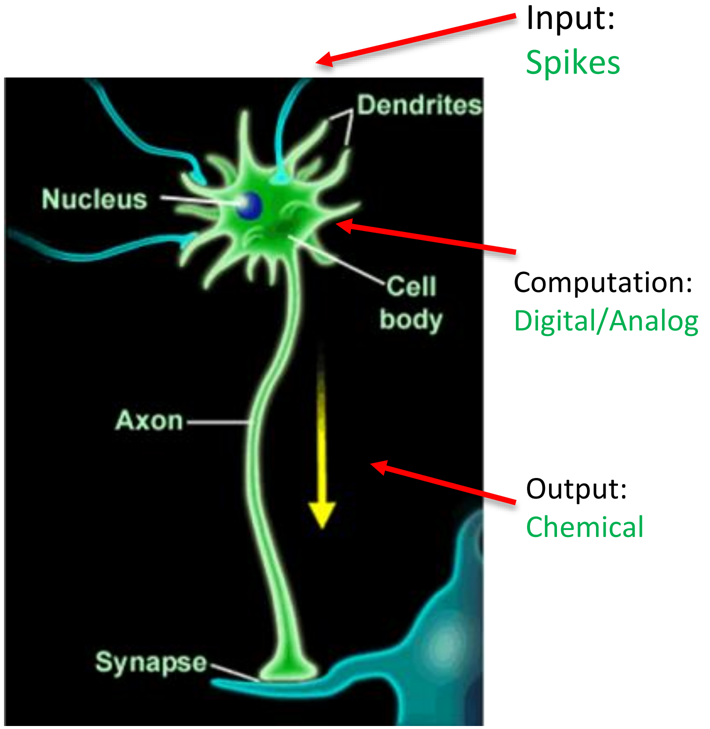


Figure 5: Neuron Structure

Outline

- Background
- Motivation
- Paper Contribution
- System Architecture
- Preliminary Evaluation
- Conclusion and Future work

Paper Contribution

- Design and evaluation of an SRAM Based Spiking Neural Network System for Traffic-Light Recognition in Autonomous Vehicles:
  - Implementation in Verilog HDL and prototyping with FPGA
  - Evaluate the accuracy, execution time, power consumption and complexity of the system.

Outline

- Background
- Motivation
- Paper Contribution
- **System Architecture**
- Preliminary Evaluation
- Conclusion and Future work

System Architecture

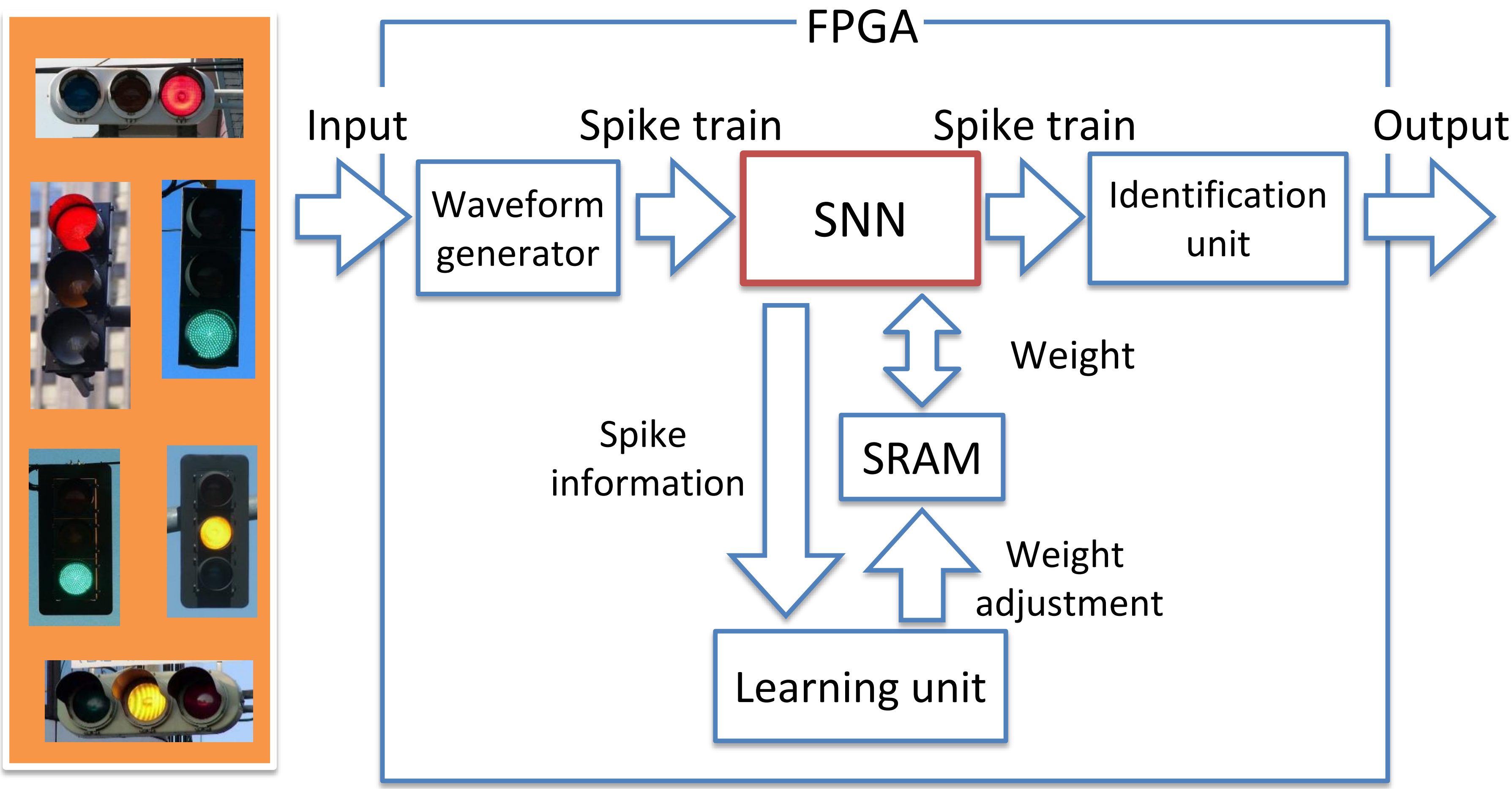


Figure 6: Spiking Neural Network System for Traffic-Light Recognition in Autonomous Vehicles.

Leaky Intergrade and Fire Neuro-Core

- Spiking Neuron:
  - Input and output is event.
  - Has Internal potential and threshold.
  - When the potential exceeds the threshold value, the neuron spike(fire).

Leaky-Integrated and Fire core:
- Spiking Neuron + Leak value.
- Internal potential drops with time.

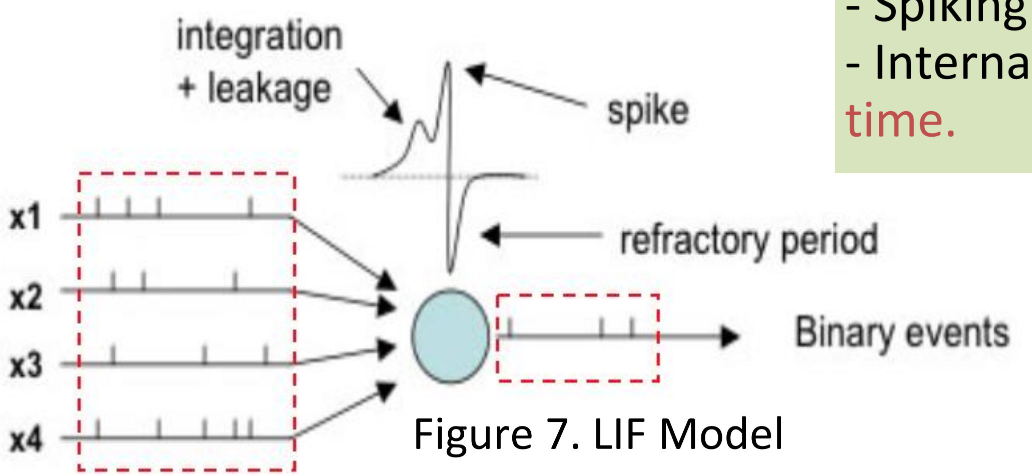


Figure 7. LIF Model

Kanta Suzuki, Yuichi Okuyama, Abderazek Ben, Abdallah, "**Hardware Design of a Leaky Integrate and Fire Neuron Core Towards the Design of a Low-power Neuro-inspired Spike-based Multicore SoC**", Information Processing Society Tohoku Branch Conference, Feb. 10, 2018.

System Training with BP

- **Training Set**
  A collection of input-output patterns that are used to train the network
- **Testing Set**
  A collection of input-output patterns that are used to assess network performance
- **Learning Rate-η**
  A scalar parameter, analogous to step size in numerical integration, used to set the rate of adjustments

System Training with BP: Network Error

- Total-Sum-Squared-Error (TSSE)

$$TSSE = \frac{1}{2} \sum_{patterns} \sum_{outputs} (Desired - Actual)^2$$

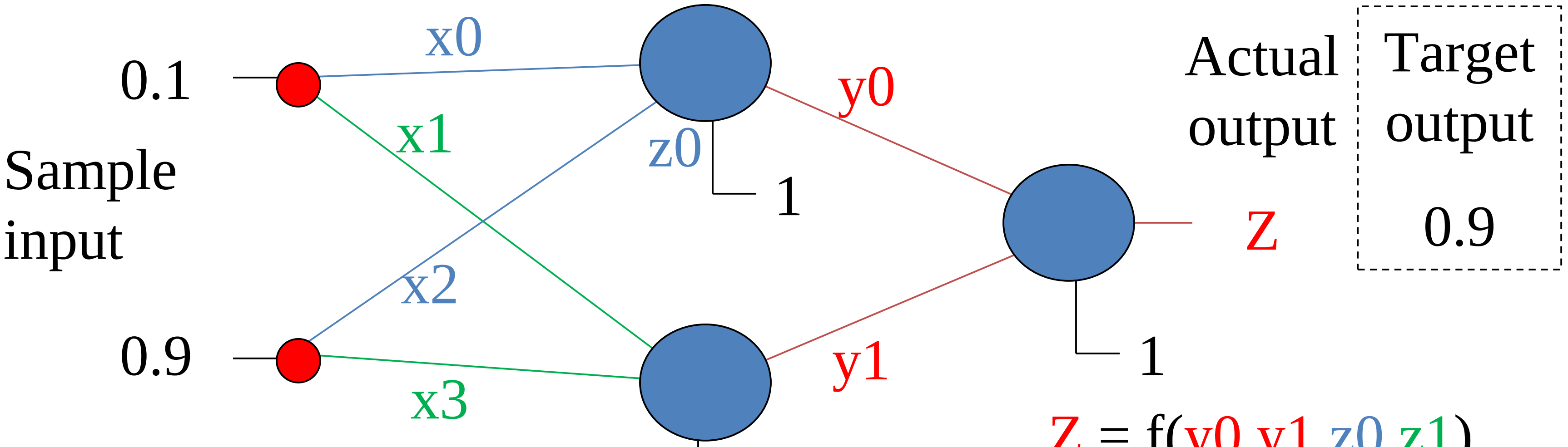
- Root-Mean-Squared-Error (RMSE)

$$RMSE = \sqrt{\frac{2 * TSSE}{\# patterns * \# outputs}}$$

System Training with BP: Pseudo-Code Algorithm

- Randomly choose the initial weights
- While error is too large
  - For each training pattern (presented in random order)
    - Apply the inputs to the network
    - Calculate the output for every neuron from the input layer, through the hidden layer(s), to the output layer
    - Calculate the error at the outputs
    - Use the output error to compute error signals for pre-output layers
    - Use the error signals to compute weight adjustments
    - Apply the weight adjustments
  - Periodically evaluate the network performance

System Training with BP: Example



Z = f(y0,y1,z0,z1)

z0 = f(x0,x2)

z1 = f(x1,x3)

Z = f(y0,y1,f(x0,x2),f(x1,x3))

Figure 7. BP Example

System Hardware Design

```
-- first load bias weight into sum
WHEN hidden_weighted_bias_load =>
   sram_mode <= read;
   sram_addr <= std_logic_vector(to_unsigned
   state <= hidden_weighted_bias_load_comple
WHEN hidden_weighted_bias_load_complete =>
   hidden_outputs(h) <= sram_output;
   i <= 0;
   state <= hidden_weighted_value_load;
```

Fig.8Code for Load bias and weight.

```
-- sum up all weighted input to the hidden
WHEN hidden_weighted_value_load =>
   sram_mode <= read;
   sram_addr <= std_logic_vector(to_unsign
   state <= hidden_weighted_value_load_com
WHEN hidden_weighted_value_load_complete =
   weight <= sram_output;
   state <= hidden_weighted_value_mul;
WHEN hidden_weighted_value_mul =>
```

Fig8.Code for Forward propagation.

Fig8.Code for Sigmoid function.

```
-- start sigmoid calculation
WHEN hidden_sig_neg =>
   -- sum = -sum
   hidden_outputs(h)(31) <= not hidden_outpu
   state <= hidden_sig_exp;
WHEN hidden_sig_exp =>
   -- output = exp(-sum)
   float_alu_a <= hidden_outputs(h);
   float_alu_mode <= exp;
   state <= hidden_sig_exp_complete;
WHEN hidden_sig_exp_complete =>
```

System Hardware Design

```
-- output layer error
WHEN output_err_sub =>
   -- error = target - output
   float_alu_a <= targets(o);
   float_alu_b <= output_outputs(o);
   float_alu_mode <= sub;
   state <= output_err_sub_complete;
WHEN output_err_sub_complete =>
   f <= float_alu_a;
```

Fig8.Code for Calculation network error.

```
-- calculate delta for output layer
WHEN output_delta_sub =>
   -- delta = 1.0 - output
   float_alu_a <= float_one;
   float_alu_b <= output_outputs(o);
   float_alu_mode <= sub;
   state <= output_delta_sub_complete;
WHEN output_delta_sub_complete =>
   output_deltas(o) <= float_alu_c;
```

Fig8.Code for Calculation delta.

Fig8.Code for Parameter update.

```
-- update weight of each input connecti
WHEN output_update_weight_mul =>
   -- alpha * delta * connection value
   float_alu_a <= a;
   float_alu_b <= hidden_outputs(h);
   float_alu_mode <= mul;
   state <= output_update_weight_mul_cc
WHEN output_update_weight_mul_complete
   f <= float_alu_c;
   state <= output_update_weight_load;
```

Outline

- Background
- Motivation
- Paper Contribution
- System Architecture
- Preliminary Evaluation
- Conclusion and Future work

Preliminary Performance Evaluation

Implementation of detecting 16 patterns from 16 inputs with BP.

Device: EP2C35F672C6
Family: Cyclone2
Synthesis: Quartus2 13.1

Table 1 : ANN Performance Evaluation

ALUs	Registers	Pins	Fmax
10,989 (33%)	5,814 (18%)	432 (89%)	76.02 MHz
Memory	DSP Block	Power Consumption	
4,956 (1%)	54 (77%)	286.84 mW	

Floating point calculator One implementation uses 77% of DPS. In conventional ANN Implementation, its calculation can not be performed in parallel, and it is difficult to calculate large-scale ANN in real time.

Outline

- Background
- Motivation
- Paper Contribution
- System Architecture
- Preliminary Evaluation
- Conclusion and Future work

Conclusion and Future Work

- This paper presented architecture and preliminary evaluation of an SRAM Based Neural Network System for Traffic-Light Recognition in Autonomous Vehicles.
- Future work will focus on Hardware implementation and evaluation of SNN model for Traffic-Light Recognition in Autonomous Vehicles.

Thank you for your Attention.