Seminar Nasional Ilmu Teknik dan Aplikasi Industri (SINTA)

Homepage: sinta.eng.unila.ac.id

# Membangun Rangkaian Kombinasional Banyak Menggunakan Struktur Jaringan Neural Tunggal

Ageng S. Repelianto[a,*], Akbar Hidayat[a], Herlinawati[a]
[a]*Jurusan Teknik Elektro, Fakultas Teknik, Universitas Lampung, Jl. Prof. Sumantri Brojonegoro No.1 Bandar Lampung, 35145*
**ageng.sadnowo@eng.unila.ac.id*

INFORMASI ARTIKEL

*Riwayat artikel:*
Diterima 12/11/2025
Direvisi 10/12/2025

*Kata kunci:*
Rangkaian kombinasional
Jaringan syaraf tiruan tunggal
Metode *Backpropagation*
*Database* kelompok bobot
Mikrokontroller
Algoritma pengaktifan set bobot

ABSTRAK

Perangkat keras digital adalah bagian penting dari sistem-sistem elektronik berbasis digital. Sistem pensaklaran digital umumnya dibangun dengan rangkaian kombinasional ataupun rangkaian sekuensial. Rangkaian kombinasional unggul pada respon cepat sementara rangkaian sekuensial lebih minimalis dalam penggunaan hardware. Fungsi kerja dari suatu rangkaian kombinasional dinyatakan dalam fungsi Boolean. Untuk setiap fungsi yang unik dari rangkaian kombinasional, maka secara *hardware* susunannya perlu disusun khas tersendiri. Sebagai konsekuensi, ketika dibutuhkan banyak rangkaian kombinasional yang berbeda, maka sebanyak itu pula komponen untuk rangkaian *hardware* yang dibutuhkan, dan ini menjadi mahal. Pada penelitian ini diusulkan penggunaan sebuah arsitektur jaringan syaraf tiruan (JST) tunggal untuk memodelkan tiga buah rangkaian kombinasional uji (RKU) yang berbeda fungsi booleannya dengan jumlah input dan output yang sama. Pembelajaran JST menggunakan metode *backpropagation*. Gagasan baru yang diusulkan adalah membuat *database* bobot-bobot JST yang dikelompokkan dalam tiga set kelompok pembobotan sesuai dengan fungsi Boolean masing-masing RKU. Ketika arsitektur JST difungsikan sebagai rangkaian RKU1 maka set pembobotan RKU1 menjadi nilai-nilai bobot aktif pada JST. Demikian juga untuk fungsi Boolean RKU2 dan RKU3. Set pembobotan untuk RKU2 atau RK3 di set aktif pada JST saat diinginkan. Arsitektur JST dibangun menggunakan mikrokontroller Arduino. Hasil pengujian menunjukan bahwa algoritma pengaktifan set pembobotan fungsi rangkaian RKU1, RKU2 atau RKU3 berfungsi sesuai saat dipanggil sebagai nilai JST. Keberhasilan ini memberikan banyak ide dalam membangun fungsi pensaklaran kombinasional menggunakan JST yang menjanjikan efisiensi perangkat keras dan lebih murah.

## 1. Pendahuluan

Perkembangan JST telah banyak dicoba untuk menggantikan fungsi kerja dari model suatu rangkaian elektronik. Dalam memodelkan karakteristik resistif non linier agar dapat bersikap sebagai resistor linier dalam rangkaian Chua, model JST berhasil merespon dengan baik dalam simulator prilaku yang diharapkan (Andrejevic Stosovic, Miona & Litovski, Vanco 2003). Penerapan JST dalam aktifias diagnosis gangguan katastropik pada model matematik rangkaian analog linier mampu menunjukkan kinerja yang baik pada semua fase diagnosa. Hal yang sama juga dilakukan diagnosa pada rangkaian digital yang diterapkan pada FPGA. JST berhasil mengurangi kebutuhan perangkat keras [1], [2]. Pada penelitian lain, JST berbasis FPGA menunjukkan kinerja yang sesuai pada teknik optimasi Unit Logika Aritmatika (ALU) variabel presisi. Teknik ini mampu meningkatkan troughput komputasi dan efisiensi energi serta latensi oleh pemrosesan JST [2].

Dalam diagnosis kesalahan perangkat keras pada sirkuit digital, JST diterapkan untuk mendiagnosis

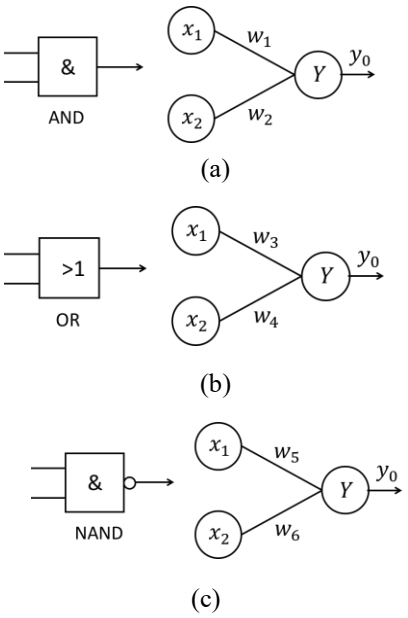
berbagai kesalahan pada rangkaian. Sejumlah data uji rangkaian dari tabel kebenaran kesalahan tunggal stuck-at membentuk pohon diagnosa uji. Prosedur diagnosa berbasis JST berhasil dicapai untuk efisiensi pengujian [3]. Pada kasus diagnosa rangkaian digital lain, JST back propagation dilatih untuk memvalidasi kesesuaian kurva polynomial untuk klasifikasi kesalahan. Kemudian menggunakan instrumen virtual untuk mengimplementasikan sistem diagnosis kesalahan tersebut. Keluaran rangkaian yang diuji, dalam kondisi tanpa kesalahan atau memiliki kesalahan. Eksperimen pada gerbang universal, untuk semua kesalahan yang disimulasikan berhasil didiagnosa dan divalidasi dengan benar [4]. Penelitian serupa dilakukan oleh Virendra. Model matematika untuk sistem deteksi kesalahan rangkaian eksternal diimplementasikan menggunaan JST pada FPGA. Langkah ini mampu mengurangi kebutuhan hardware untuk pengali dan pembagi. Hasil sistem, secara logis serupa dengan perbandingan rangkaian uji eksternal[5]. Pola yang sama penggunaan JST untuk mendiagnosa gangguan kesalahan dan klasifikasi kesalahan pada pemantauan proses dalam rekayasa sistem proses. JST menunjukkan keunggulan dibandingkan dengan metode konvensional berbasis data [6]. Demikian juga penerapan diagnostik kesalahan dalam sistem Pelacakan foto sel yang mengalami kerusakan. JST dapat menemukan lokasi foto sel yang rusak mengamati pola karakteristik arus-tegangan (IeV) dari modul fotovoltaik (PV) [7].

Pada ranah implementasi perangkat keras untuk JST, sistem digital adalah yang paling mudah untuk mewujudkannya [8]. Penggunaan FPGA memberikan fleksibilitas dalam sistem yang dapat diprogram. Sehingga, prototipe instrumen berbasis JST dalam aplikasi waktu nyata, desain chip saraf FPGA memiliki kecepatan lebih tinggi dan ukuran lebih kecil dari pada desain menggunakan VLSI [9], [10]. Namun demikian, realisasi FPGA sebagai JST dengan sejumlah besar neuron masih merupakan tugas yang menantang [11].

Konsep dari proses diagnosa suatu fungsi sirkuit digambarkan oleh pohon diagnosa yang tersusun sedemikian rupa membentuk himpunan nilai-nilai pengujian. Pembangkitan urutan nilai uji memiliki proses seperti fungsi sebuah rangkaian kombinasional. Pola inilah yang kemudian meniscayakan JST dibangun untuk fungsi suatu rangkaian kombinasional. Dalam karya ini disajikan kontribusi penggunaan sebuah arsitektur JST yang berperan untuk banyak fungsi rangkaian digital kombinasional. Gagasan ini mengusulkan sebuah data base berisi beberapa kelompok bobot jaringan dari JST. Setiap kelompok dapat dipanggil untuk berfungsi sebagai rangkaian digital yang dibutuhkan. Sebagai syarat dari arsitektur ini adalah semua rangkaian digital kombinasional tersebut memiliki jumlah input dan output yang sama.

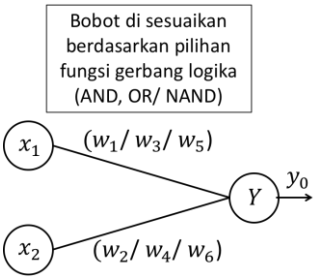
### *1.1. Konsep Arsitektur JST untuk Banyak Rangkaian Kombinasional*

Secara sederhana konsep JST untuk banyak rangkaian kombinasional seperti ditunjukkan pada Gambar 1. Terdapat tiga buah fungsi gerbang logika AND, OR, dan NAND yang diimplementasikan oleh JST.




**Gambar 1.** JST untuk banyak rangkaian kombinasional. (a) JST gerbang AND, (b) JST gerbang OR, (c) JST gerbang NAND,

Ketiga JST, memiliki arsitektur, jumlah input dan output yang serupa. Perbedaannya terletak pada nilai bobot pada masing-masing JST untuk fungsi yang berbeda. Bobot ($w_1$, $w_2$) untuk gerbang AND, bobot ($w_3$, $w_4$) dan bobot ($w_5$, $w_6$) berturut-turut untuk gerbang OR dan NAND.




**Gambar 2.** Konsep sebuah arsitektur JST untuk banyak rangkaian kombinasional.

Berdasarkan pola ini, gagasan yang diajukan adalah dengan sebuah arsitektur JST dimungkinkan untuk difungsikan sebagai banyak RKU lain yang memiliki keserupaan arsitektur. Bobot-bobot pada hubungan neuron yang sama dikelompokkan. Kemudian, kelompok bobot dipilih untuk dipasang pada jaringan sesuai kebutuhan saat fungsi RKU dibutuhkan. Konsep

ini diilustrasikan seperti Gambar 2.

### *1.2. JST Backpropagation*

Metode JST *Backpropagation* memiliki kelebihan nilai Root Mean Square Error (RMSE) yang rendah dengan akurasi yang lebih baik meskipun diberikan data latih 50 persen dari data total. Kelebihan ini menjadi alasan pilihan dalam penerapan JST. Berikut ini adalah persamaan-persamaan yang digunakan dalam proses pembelajaran *Backpropagation* [12], [13].

1. Fungsi aktifasi yang digunakan adalah *sigmoid* yang memiliki rentang 0 sampai 1 seperti persamaan 1.

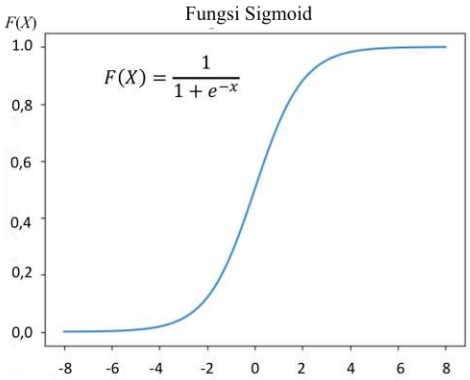
$$y = f(x) = \frac{1}{1 + e^{-x}} \quad (1)$$

dengan,

$$X = b + \sum x_{ij} w_i \quad (2)$$

dimana;

$X$ adalah kombinasi linier dari input, bobot dan bias.
$W$ adalah bobot
$X$ adalah input neuron
$b$ adalah *Bias*




**Gambar 3.** Fungsi Sigmoid.

2. Turunan dari *Sigmoid*

$$f'(x) = \sigma(x)[1 - \sigma(x)] \quad (3)$$

3. *Forward Propagation*

$$y_o = f\left(b + \sum x_{ij} w_i\right) \quad (4)$$

4. Menghitung *Mean Squared Error*

$$fE = \frac{1}{2} x (y_{Target} - y_{pred})^2 \quad (5)$$

Dimana,

$y_{Target}$ adalah target *output*,
$y_{pred}$ adalah prediksi *output* fungsi *sigmoid*.

5. Pembaruan Bobot dan *Bias*

$$w_{baru} = w_{lama-}\eta . \frac{\partial E}{\partial w} \quad (6)$$

$$b_{baru} = b_{lama-}\eta . \frac{\partial E}{\partial b} \quad (7)$$

Keterangan:
$\eta$ = *learning rate*

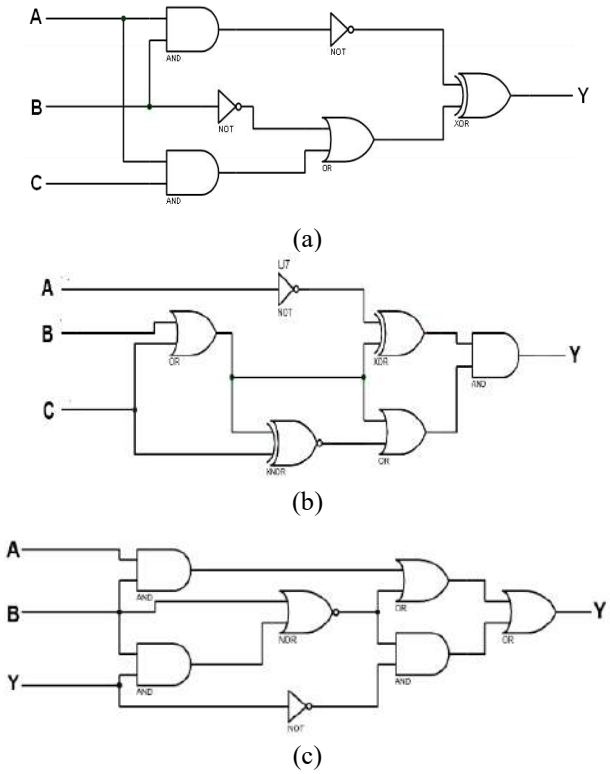
$\frac{\partial E}{\partial b}$ = gradien dari fungsi *error* terhadap *bias*

$\frac{\partial E}{\partial w}$ = gradien dari fungsi *error* terhadap bobot.

## 2. Metode Penelitian

### *2.1. Rangkaian Kombinasional Uji (RKU)*

Rangkaian kombinasional uji yang digunakan ada sebanyak tiga rangkaian seperti pada _Gambar 4. Ketiga RKU memiliki kesamaan jumlah input tiga buah dengan sebuah output.




**Gambar 4.** Rangkaian Kombinasional Uji, a) RKU1, b) RKU2, c) RKU3.

RKU 1, RKU 2 dan RKU 3, secara berturut-turut mempunyai persamaan fungsi Boolean sebagai berikut;

$$Y1 = (\overline{A.B}) \oplus \left(\bar{B} + (A.C)\right) \quad (8)$$

$$Y2 = \left(\bar{A} \oplus (B + C)\right).\left((B + C) + \overline{(\mathrm{B} + \mathrm{C}) \oplus \mathrm{C}}\right) \quad (9)$$

$$Y3 = \left((A.B) + \left(\overline{B + (B.C)}\right)\right) + \left((\overline{B + (B.C)}\,.\,\bar{C}\right) \quad (10)$$

Sementara itu, hubungan input dan output mempunyai karakteristik respon fungsi Boolean ditunjukan pada Tabel 1.

**Tabel 1.** Tabel kebenaran; (a) RKU 1, (b) RKU 2, (c) RKU 3.

| Input | | | Output |
|---|---|---|---|
| A | B | C | Y1 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

(a)

| Input | | | Output |
|---|---|---|---|
| A | B | C | Y1 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

(b)

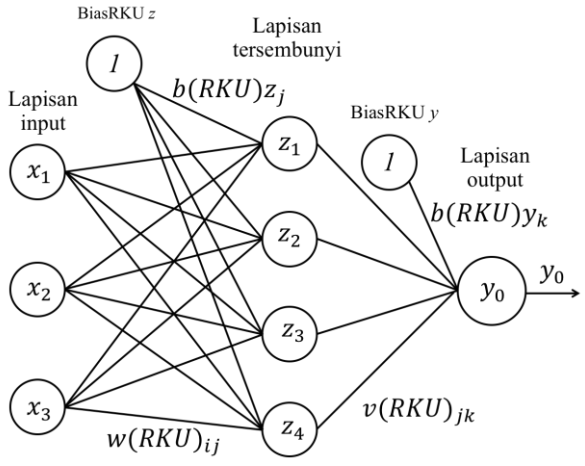
| Input | | | Output |
|---|---|---|---|
| A | B | C | Y1 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

(c)

Ketiga RKU tidak mempermasalahkan jika tersusun dari sejumlah tingkat proses yang berbeda. RKU1 dan RKU2 terdiri dari tiga tingkat proses, sementara RKU3 terdiri dari empat tingkat proses.

## *2.2. Arsitektur JST RKU*

Dalam kajian ini, arsitektur JST menggunakan multi layer dengan tiga neuron input, empat neuron hidden, dengan setiap neuron hidden ditambahkan sebuah bias.




**Gambar 5.** Arsitektur JST uji.

Selanjutnya pada neuron output juga ditambahkan sebuah bias output. Tujuan penambahan bias sebagai penegas area nol, pemisah antara nilai negatif dan nilai positif dari output. Usul arsitektur ini ditunjukkan pada Gambar 5.

Arsitektur ini menyesuaikan dengan rangkaian kombinasional uji dengan tiga input dan sebuah output seperti yang disajikan pada Gambar 4.

## *2.3. Variabel bobot arsitektur JST*

Penamaan variabel bobot dari ketiga RKU disampaikan dalam Tabel 2. Bobot $w(\text{RKU})_{xz}$ artinya bobot dari RKU yang dipilih pada posisi di jaringan antara neuron *x* ke neuron *z*.

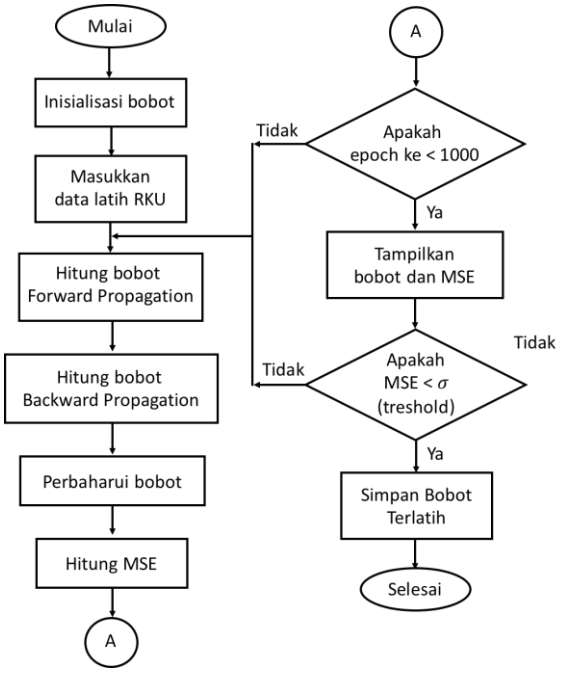
**Tabel 2.** Variabel pembobotan JST RKU.

| RKU | Bobot neuron | | | | | |
|---|---|---|---|---|---|---|
| | $x_i$ ke $z_j$ ($w$(RKU) $_{xz}$) | | | Bias ke $z_j$ ($b$(RKU) $_z$) | $z_j$ ke $y_o$ $v$(RKU) $_{zy}$ | Bias ke $y_o$ $b$(RKU)$y$ |
| RKU 1 | $w1_{11}$ | $w1_{21}$ | $w1_{31}$ | $b1z_1$ | $v1_{10}$ | $b1y$ |
| | $w1_{12}$ | $w1_{22}$ | $w1_{32}$ | $b1z_2$ | $v1_{20}$ | |
| | $w1_{13}$ | $w1_{23}$ | $w1_{33}$ | $b1z_3$ | $v1_{30}$ | |
| | $w1_{14}$ | $w1_{24}$ | $w1_{34}$ | $b1z_4$ | $v1_{40}$ | |
| RKU 2 | $w2_{11}$ | $w2_{21}$ | $w2_{31}$ | $b2z_1$ | $v2_{10}$ | $b2y$ |
| | $w2_{12}$ | $w2_{22}$ | $w2_{32}$ | $b2z_2$ | $v2_{20}$ | |
| | $w2_{13}$ | $w2_{23}$ | $w2_{33}$ | $b2z_3$ | $v2_{30}$ | |
| | $w2_{14}$ | $w2_{24}$ | $w2_{34}$ | $b2z_4$ | $v2_{40}$ | |
| RKU 3 | $w3_{11}$ | $w3_{21}$ | $w3_{31}$ | $b3z_1$ | $v3_{10}$ | $b3y$ |
| | $w3_{12}$ | $w3_{22}$ | $w3_{32}$ | $b3z_2$ | $v3_{20}$ | |
| | $w3_{13}$ | $w3_{23}$ | $w3_{33}$ | $b3z_3$ | $v3_{30}$ | |
| | $w3_{14}$ | $w3_{24}$ | $w3_{34}$ | $b3z_4$ | $v3_{40}$ | |

*Keterangan:*
$w1_{ij}$ dimana;
*w* adalah bobot antara neuron $x_i$ ke $z_j$
1 adalah RKU1
*i* adalah neuron *x* dengan *i* = 1, 2, …, n
*j* adalah neuron *z* dengan *j* = 1, 2, …, m

## *2.4. Menghitung variabel bobot RKU dengan metode backpropagation*

Tahap berikut ini, adalah melakukan proses pembelajaran untuk menentukan nilai variabel pembobotanRKU1, RKU2 dan RKU3 sebagaimana tabel 2. Pembelajaran dipilih menggunakan metode *backpropagation* dengan proses komputasi sebagaimana diagram alir pada gambar 4. [12]




**Gambar 6.** Diagram alir proses pembelajaran JST RKU menggunakan metode *backpropagation*.

Pada proses pembelajaran untuk penentuan bobot jaringan semua RKU seperti Tabel 2, perhitungan pada algoritma menggunakan persamaan (1) sampai dengan persamaan (7). Sebagaimana prasarat di awal, jumlah input dan jumlah output RKU harus sama. Bobot dihitung menggunakan metode pembelajaran yang sama dengan fungsi aktifasi, jenis input dan output, *learning rate* dan *threshold* yang juga sama.

## *2.5. Algoritma pemilihan fungsi RKU*

Bagian penting dari algorima JST ini adalah fungsi pemilihan RKU yang berperan melakukan *setting* bobot jaringan neuron disesuaikan dengan fungsi RKU yang dipilih. Gagasan fungsi pemilihan inilah yang menjadi kepraktisan untuk implementasi banyak rangkaian kombinasional, tanpa harus menyusun arsitektur JST baru sebagai fungsi yang berbeda-beda.

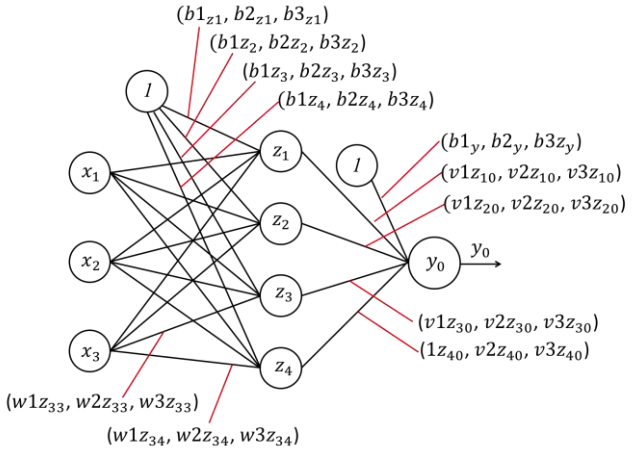
Kelompok bobot RKU terdiri dari dua bagian. Pertama bobot dari neuron lapisan input $x_i$ ke neuron lapisan tersembunyi $z_j$ dengan variabel $w(\text{RKU})_{ij}$ dan sebuah neuron bias $b(\text{RKU})_{zj}$ untuk setiap neuron $z_j$.

Bagian kedua, bobot dari neuron lapisan input $z_j$ ke neuron output $y_o$ dengan variabel $v(\text{RKU})_{jy}$ dan sebuah neuron bias $b(\text{RKU})_y$. Tabel 3 menunjukkan semua variabel dari JST.

**Tabel 3.** Kelompok variabel bobot semua RKU pada setiap hubungan neuron.

| Pilih PKU | Bobot neuron | | | | | |
|---|---|---|---|---|---|---|
| | $x_i$ ke $z_j$ ($w(\text{RKU})_{xz}$) | | | Bias ke $z_i$ ($b(\text{RKU})_z$) | $z_j$ ke $y_o$ $v(\text{RKU})_{zy}$ | Bias ke $y_o$ $b(\text{RKU})y$ |
| (RKU1/ RKU2/ RKU3) | ($w1_{11}$/ $w2_{11}$/ $w3_{11}$) | ($w1_{21}$/ $w2_{21}$/ $w3_{21}$) | ($w1_{31}$/ $w2_{31}$/ $w3_{31}$) | ($b1z_1$/ $b2z_1$/ $b3z_1$) | ($v1_{10}$/ $v2_{10}$/ $v3_{10}$) | ($b1y$/ $b2y$/ $b3y$) |
| | ($w1_{12}$/ $w2_{12}$/ $w3_{12}$) | ($w1_{22}$/ $w2_{22}$/ $w3_{22}$) | ($w1_{32}$/ $w2_{32}$/ $w3_{32}$) | ($b1z_2$/ $b2z_2$/ $b3z_2$) | ($v1_{20}$/ $v2_{20}$/ $v3_{20}$) | |
| | ($w1_{13}$/ $w2_{13}$/ $w3_{13}$) | ($w1_{12}$/ $w2_{12}$/ $w3_{12}$) | ($w1_{22}$/ $w2_{23}$/ $w3_{22}$) | ($b1z_3$/ $b2z_3$/ $b3z_3$) | ($v1_{30}$/ $v2_{30}$/ $v3_{30}$) | |
| | ($w1_{14}$/ $w2_{14}$/ $w3_{14}$) | ($w1_{24}$/ $w2_{24}$/ $w3_{24}$) | ($w1_{34}$/ $w2_{34}$/ $w3_{34}$) | ($b1z_4$/ $b2z_4$/ $b3z_4$) | ($v1_{40}$/ $v2_{40}$/ $v3_{40}$) | |

Sementara itu, penerapan kelompok atau himpunan bobot fungsi RKU, diilustrasikan pada Gambar 7.




**Gambar 7.** Kelompok pembobotan pada jaringan antar neuron arsitektur JST RKU.

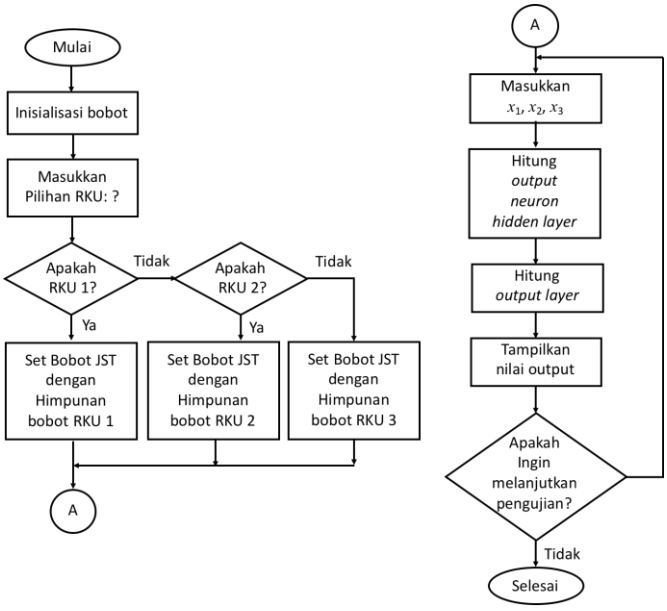
Selanjutnya, algoritma pemanggilan kelompok bobot sesuai keperluan fungsi RKU disajikan seperti diagram alir Gambar 8.

Pada algoritma terdapat fungsi *test gate_type* yang digunakan untuk menguji gerbang logika berdasarkan tipe RKU yang dipilih. Fungsi ini memeriksa nilai input *gate_type* dan memilih bobot serta *bias* yang sesuai untuk kombinasi gerbang logika yang diinginkan. Jika nilai *gate_type* adalah RKU1, maka bobot dan *bias* RKU1 yang pasangkan. Demikian pula untuk RKU2 dan RKU3. Fungsi ini memproses program untuk menguji berbagai RKU menggunakan bobot dan *bias* yang telah dilatih sebelumnya.

Kode untuk seleksi bobot disetiap RKU seperti pada daftar kode berikut:

```
# Fungsi kelompok pembobotan RKU
def test_gate(gate_type):
     """Menguji gerbang logika menggunakan
bobot yang sudah dilatih."""
    if gate_type == "kombinasi 1":
       weights_input_hidden,
       weights_hidden_output=
```

```
            weights_kombinasi_1 Bias_hidden,
            Bias_output = Bias_kombinasi_1
        elif gate_type == "kombinasi 2":
            weights_input_hidden,
            weights_hidden_output=
            weights_kombinasi_2 Bias_hidden,
            Bias_output = Bias_kombinasi_2
        elif gate_type == "kombinasi 3":
            weights_input_hidden,
            weights_hidden_output=
            weights_kombinasi_3
            Bias_hidden, Bias_output =
            Bias_kombinasi_3
        else:
            print("Gerbang logika tidak valid.")
            return
    all_inputs_outputs = []
        all_raw_data = [] # Menyimpan data
        mentah
```




**Gambar 8.** Diagram alir algoritma pemilihan fungsi RKU.

## 3. Hasil Dan Pembahasan

### *3.1. Komputasi bobot JST dengan backpropagation*

Hasil pelatihan *backpropagation* memberikan hasil akhir pembobotan arsitektur JST gambar 2, sesuai masing-masing RKU diberikan pada Tabel 4, 5 dan 6.

**Tabel 4.** Tabel bobot antara neuron untuk RKU1.

| Neuron | $z_1$ | $z_2$ | $z_3$ | $z_4$ |
|---|---|---|---|---|
| $x_1$ | 0,985 | 1,932 | 4,090 | 2,060 |
| $x_2$ | -0,873 | -1,571 | -0,570 | 0,104 |
| $x_3$ | 0,906 | 4,245 | 4,742 | 0,678 |
| $bz$ | 0,897 | -3,023 | 1,065 | 0,978 |
| $y_o$ | -2,324 | 6,515 | -0,653 | -0,846 |
| $by_o$ | | -1,249 | | |

**Tabel 5.** Tabel bobot antara neuron untuk RKU2.

| Neuron | $z_1$ | $z_2$ | $z_3$ | $z_4$ |
|---|---|---|---|---|
| $x_1$ | 2,879 | 7,612 | 0,930 | 0,902 |
| $x_2$ | 2,825 | -3,368 | -3,575 | 0,243 |
| $x_3$ | 1,430 | 1,798 | -3,232 | -1,792 |
| $bz$ | -0,975 | 0,271 | 0,713 | 2,335 |
| $y_o$ | -5,786 | -2,646 | -0,540 | 7,593 |
| $by_o$ | | -1,401 | | |

**Tabel 6.** Tabel bobot antara neuron untuk RKU 3.

| Neuron | $z_1$ | $z_2$ | $z_3$ | $z_4$ |
|---|---|---|---|---|
| $x_1$ | 6,098 | 0,798 | -3,240 | 0,866 |
| $x_2$ | 0,791 | 1,264 | 6,370 | 0,113 |
| $x_3$ | 3,648 | 6,547 | 0,258 | 0,680 |
| $b_z$ | 0,600 | 1,021 | 1,137 | 0,221 |
| $y_o$ | 0,600 | 1,021 | 1,137 | 7,593 |
| $b_{yo}$ | | -0,830 | | |

Tingkat pertama adalah nilai bobot antara neuron *input layer* $x_i$, *hidden layer* $z_j$ dan bias_$b_z$ . Tingkat ke 2 adalah nilai bobot antara neuron *hidden layer* $z_j$, bias output _$b_{yo}$ dan output $y_o$. Tabel 4, 5 dan 6 ini juga disebut sebagai himpunan bobot masing-masinng RKU.

### *3.2. Hasil Pengujian JST Dengan Bobot Terlatih*

Pengujian komputasi JST dilakukan dengan memberikan variasi input seperti tabel 2, secara berturut-turut merupakan fungsi kombinasional dari RKU1, RKU2 dan RKU3. Input dan output ditetapkan biner, dengan *treshold* $\sigma$ ditetapkan 0,5, serta fungsi aktivasi *sigmoid*. Gambar 7 adalah hasil komputasi algoritma dengan pemasangan kelompok bobot RKU 1 Tabel 4 pada JST.

```
RKU pilih: RKU 1
Input Xi: [0, 0, 0] - Output: 0
Input Xi: [0, 0, 1] - Output: 0
Input Xi: [0, 1, 0] - Output: 1
Input Xi: [0, 1, 1] - Output: 1
Input Xi: [1, 0, 0] - Output: 0
Input Xi: [1, 0, 1] - Output: 0
Input Xi: [1, 1, 0] - Output: 0
Input Xi: [1, 1, 1] - Output: 1
Hasil JST:
Input: [0, 0, 0] - Hasil yo: [-3.698498] -> threshold->: 0.5-> Hasil akhir: 0
Input: [0, 0, 1] - Hasil yo: [-1.949470] -> threshold->: 0.5-> Hasil akhir: 0
Input: [0, 1, 0] - Hasil yo: [2.017161] -> threshold->: 0.5-> Hasil akhir: 1
Input: [0, 1, 1] - Hasil yo: [3.413382] -> threshold->: 0.5-> Hasil akhir: 1
Input: [1, 0, 0] - Hasil yo: [-4.45354] -> threshold->: 0.5-> Hasil akhir: 0
Input: [1, 0, 1] - Hasil yo: [-3.967624] -> threshold->: 0.5-> Hasil akhir: 0
Input: [1, 1, 0] - Hasil yo: [-1.689187] -> threshold->: 0.5-> Hasil akhir: 0
Input: [1, 1, 1] - Hasil yo: [1.693247] -> threshold->: 0.5-> Hasil akhir: 1
```

Gambar 7. Hasil komputasi JST dengan kelompok bobot tabel 3 untuk RKU1.

```
RKU pilih: RKU 2
Input Xi: [0, 0, 0] - Output: 1
Input Xi: [0, 0, 1] - Output: 0
Input Xi: [0, 1, 0] - Output: 0
Input Xi: [0, 1, 1] - Output: 0
Input Xi: [1, 0, 0] - Output: 0
Input Xi: [1, 0, 1] - Output: 1
Input Xi: [1, 1, 0] - Output: 1
Input Xi: [1, 1, 1] - Output: 1
Hasil JST:
Input: [0, 0, 0] - Hasil yo: [2.073306] -> threshold->: 0.5-> Hasil akhir: 1
Input: [0, 0, 1] - Hasil yo: [-2.507950] -> threshold->: 0.5-> Hasil akhir: 0
Input: [0, 1, 0] - Hasil yo: [-2.539069] -> threshold->: 0.5-> Hasil akhir: 0
Input: [0, 1, 1] - Hasil yo: [-4.214743] -> threshold->: 0.5-> Hasil akhir: 0
Input: [1, 0, 0] - Hasil yo: [-1.572940] -> threshold->: 0.5-> Hasil akhir: 0
Input: [1, 0, 1] - Hasil yo: [2.192083] -> threshold->: 0.5-> Hasil akhir: 1
Input: [1, 1, 0] - Hasil yo: [2.212705] -> threshold->: 0.5-> Hasil akhir: 1
Input: [1, 1, 1] - Hasil yo: [2.409946] -> threshold->: 0.5-> Hasil akhir: 1
```

**Gambar 8.** Hasil komputasi JST dengan kelompok bobot Tabel 5 untuk RKU2.

```
RKU pilih: RKU 3
Input Xi: [0, 0, 0] - Output: 1
Input Xi: [0, 0, 1] - Output: 1
Input Xi: [0, 1, 0] - Output: 0
Input Xi: [0, 1, 1] - Output: 0
Input Xi: [1, 0, 0] - Output: 1
Input Xi: [1, 0, 1] - Output: 1
Input Xi: [1, 1, 0] - Output: 1
Input Xi: [1, 1, 1] - Output: 1
Hasil JST:
Input: [0, 0, 0] - Hasil yo: [2.2307201] -> threshold->: 0.5-> Output yo: 1
Input: [0, 0, 1] - Hasil yo: [2.2921629] -> threshold->: 0.5-> Output yo: 1
Input: [0, 1, 0] - Hasil yo: [-1.792248] -> threshold->: 0.5-> Output yo: 0
Input: [0, 1, 1] - Hasil yo: [-1.721798]  -> threshold->: 0.5-> Output yo: 0
Input: [1, 0, 0] - Hasil yo: [3.5853023] -> threshold->: 0.5-> Output yo: 1
Input: [1, 0, 1] - Hasil yo: [3.5088245] -> threshold->: 0.5-> Output yo: 1
Input: [1, 1, 0] - Hasil yo: [2.2445413] -> threshold->: 0.5-> Output yo: 1
Input: [1, 1, 1] - Hasil yo: [2.3438109] -> threshold->: 0.5-> Output yo: 1
```

**Gambar 9.** Hasil komputasi JST dengan kelompok bobot Tabel 6 untuk RKU3.

Demikian juga untuk Gambar 8 dan 9, merupakan hasil komputasi JST dengan berturut-turut kelompok bobot-bobot jaringan Tabel 5 dan Tabel 6.

### *3.3. Pembahasan*

Arsitektur tungal JST multilayer pada gambar 3, berhasil memodelkan fungsi RKU1, RKU2 dan juga RKU3. Bobot-bobot JST RKU1 dicapai pada nilai MSE mendekati 0,09 pada *epoh* ke 1000. Pada RKU 2 nilai MSE mendekati 0,05 pada *epoch* ke 3500. Sementara itu, bobot JST RKU 3 dicapai nilai MSE 0,01 di *epoch* 1000. Nilai-nilai MSE ini menunjukkan bahwa arsitektur JST berhasil melakukan proses pembelajaran dengan stabil.

Kemudian pada implementasi arsitektur JST, hasil yang ditunjukkan dapat berfungsi universal pada ketiga RKU, algoritma komputasi pada gambar 6, berhasil menjalankan prosesnya seperti hasil yang ditampilkan pada gambar 7, 8 dan 9. Algoritma dapat menerapkan bobot-bobot JST sesuai kebutuhan fungsi RKU yang diinginkan.

Keberhasilan arsitektur tunggal JST untuk dapat berfungsi sebagai banyak RKU memberikan alternatif pada penerapan yang lebih luas dengan tujuan efisiensi dan kepraktisan implementasi pada kompleksitas suatu sistem digital yang unik.

## 4. Kesimpulan

Usulan sebuah arsitektur JST untuk memodelkan tiga fungsi rangkaian kombinasional yang berbeda dengan persyaratan jumlah input dan jumlah outputnya sama telah berhasil dilakukan. Metode *backpropagation* memberikan nilai bobot-bobot JST stabil dengan MSE sekitar 0,01 sampai 0,09 pada *epoch* antara 1000 sampai 3500. Keberhasilan konsep ini memberikan keniscayaan wacana pada penerapan JST yang lebih luas dengan tujuan kepraktisan.

## Ucapan terima kasih

Penulis mengucapkan terima kasih kepada Lembaga Penelitian dan Pengabdian Kepada Masyarakat Universitas Lampung atas dukungan pendanaan penelitian Hibah BLU Universitas tahun 2025. Ucapan terima kasih juga disampaikan kepada staf Laboratorium Elektronika Jurusan Teknik Elektro yang telah membantu proses pengambilan data.

## Daftar Pustaka

[1] M. Andrejevic and V. Litovski, “Electronic circuits modeling using artificial neural networks,” *Journal of Automatic Control*, vol. 13, no. 1, pp. 31–37, 2003, doi: 10.2298/jac0301031a.

[2] M. Islampurkar, K. Gunale, S. Somani, and N. Bagade, “Multiple Stuck At Fault Diagnosis System For Digital Circuit On FPGA Using Vedic Multiplier and ANN,” *International Journal of Circuits, Systems and Signal Processing*, vol. 16, pp. 985–992, 2022, doi: 10.46300/9106.2022.16.120.

[3] A. A. Al-Jumah and T. Arslan, “Artificial neural network based multiple fault diagnosis in digital circuits,” in *ISCAS ’98. Proceedings of the 1998 IEEE International Symposium on Circuits and Systems (Cat. No.98CH36187)*, May 1998, pp. 304–307 vol.2. doi: 10.1109/ISCAS.1998.706919.

[4] A. Kumar and A. P. Singh, “Transistor level fault diagnosis in digital circuits using artificial neural network,” *Measurement*, vol. 82, pp. 384–390, 2016, doi: https://doi.org/10.1016/j.measurement.2015.12.045.

[5] V. V Shete, M. Islampurkar, and K. Gunale, “Optimized Mathematical Model of Digital Circuit using ANN on FPGA,” *International Journal of*

*Applied Engineering Research*, vol. 13, no. 19, pp. 14135–14141, 2018.

[6] S. Heo and J. H. Lee, "Fault detection and classification using artificial neural networks," *IFAC-PapersOnLine*, vol. 51, no. 18, pp. 470–475, 2018, doi: https://doi.org/10.1016/j.ifacol.2018.09.380.

[7] W. Chine, A. Mellit, V. Lughi, A. Malek, G. Sulligoi, and A. Massi Pavan, "A novel fault diagnosis technique for photovoltaic systems based on artificial neural networks," *Renew. Energy*, vol. 90, pp. 501–512, 2016, doi: https://doi.org/10.1016/j.renene.2016.01.036.

[8] P. Ienne and G. Kuhn, "Digital systems for neural networks," in *Digital Signal Processing Technology: A Critical Review*, P. Papamichalis and R. D. Kerwin, Eds., SPIE, 1995, p. 102790G. doi: 10.1117/12.204207.

[9] S. Sahin, Y. Becerikli, and S. Yazici, "Neural Network Implementation in Hardware Using FPGAs," in *Neural Information Processing*, I. King, J. Wang, L.-W. Chan, and D. Wang, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1105–1112.

[10] A. Dinu, M. N. Cirstea, and S. E. Cirstea, "Direct Neural-Network Hardware-Implementation Algorithm," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 5, pp. 1845–1848, May 2010, doi: 10.1109/TIE.2009.2033097.

[11] E. Z. Mohammed and H. K. Ali, "Hardware Implementation of Artificial Neural Network Using Field Programmable Gate Array," *International Journal of Computer Theory and Engineering*, vol. 5, no. 5, pp. 780–783, 2013, doi: 10.7763/ijcte.2013.v5.795.

[12] M. Buscema, "Back Propagation Neural Networks," *Subst. Use Misuse*, vol. 33, no. 2, pp. 233–270, 1998, doi: 10.3109/10826089809115863.

[13] R. Rojas, "The Backpropagation Algorithm," in *Neural Networks: A Systematic Introduction*, Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 149–182. doi: 10.1007/978-3-642-61068-4_7.