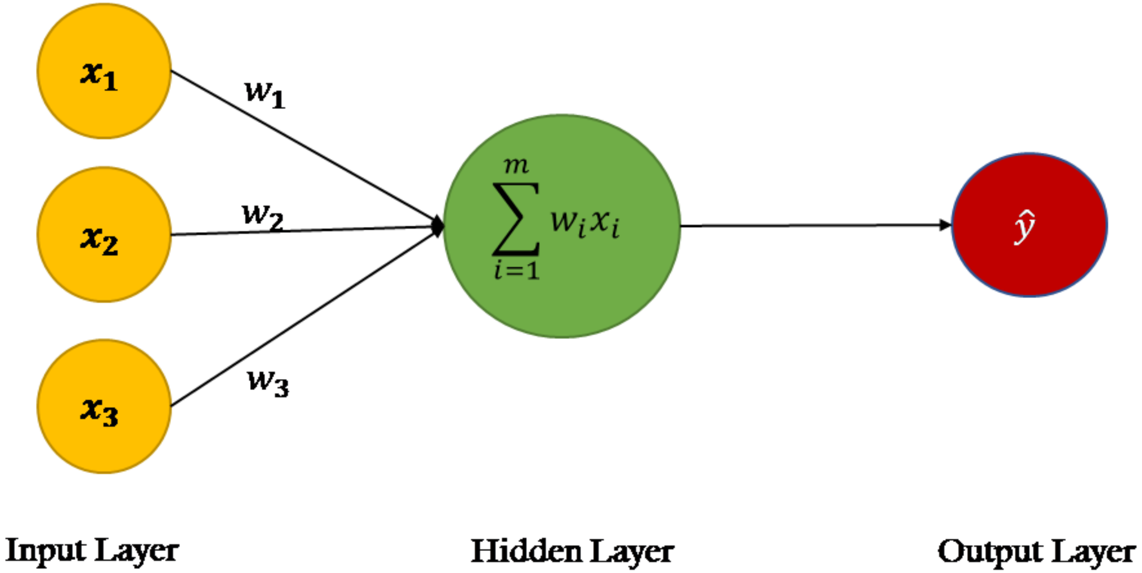
Aplikasi ANN

Erwin Setiawan

Artificial Neural Network (ANN)

Artificial Neural Network (ANN) Sederhana

- Contoh ANN sederhana dengan 3 input, 1 hidden node, dan 1 output.



y = w1*x1 + w2*x2 + w3*x3

ANN dengan Perkalian Matrix

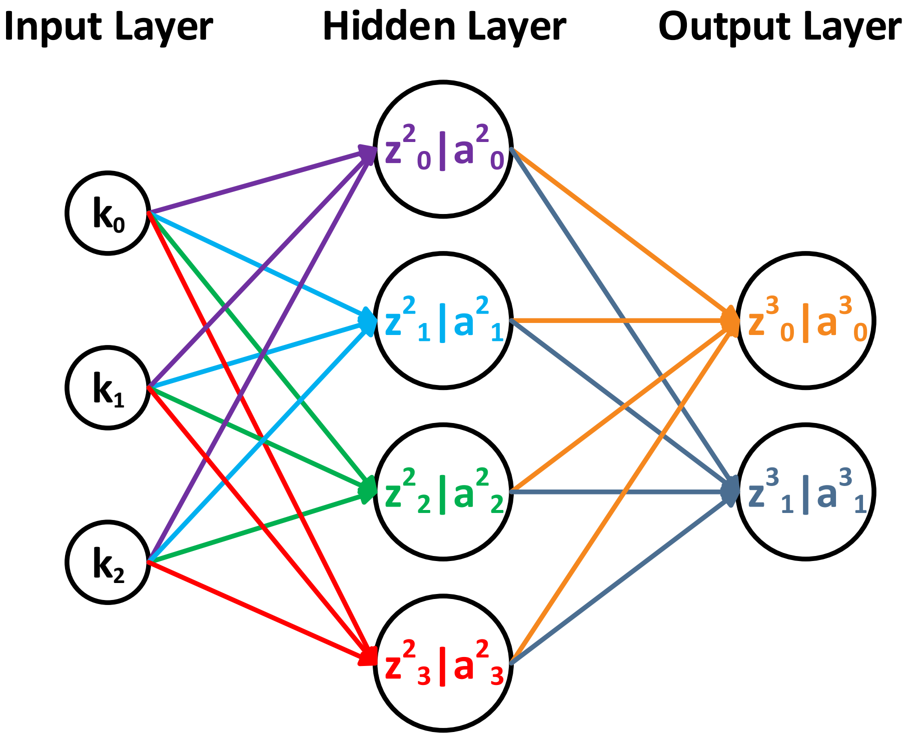
- Menghitung ANN dengan perkalian matrix. Di sini terdapat juga fungsi aktivasi sigmoid (σ).

$$\begin{bmatrix} k0 & k1 & k2 \end{bmatrix} \cdot \begin{bmatrix} w00 & w01 & w02 & w03 \\ w10 & w11 & w12 & w13 \\ w20 & w21 & w22 & w23 \end{bmatrix} = \begin{bmatrix} z0 & z1 & z2 & z3 \end{bmatrix}$$

$$\sigma(\begin{bmatrix} z0 & z1 & z2 & z3 \end{bmatrix}) = \begin{bmatrix} a0 & a1 & a2 & a3 \end{bmatrix}$$

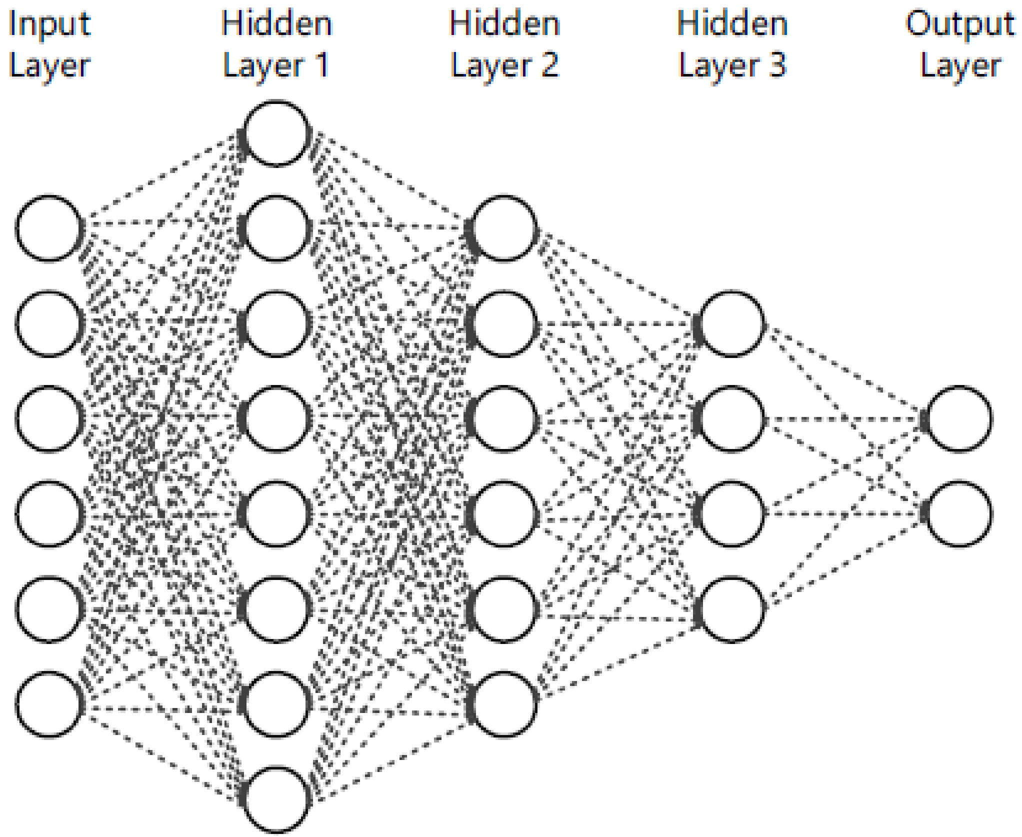
$$\begin{bmatrix} a0 & a1 & a2 & a3 \end{bmatrix} \cdot \begin{bmatrix} w00 & w01 \\ w10 & w11 \\ w20 & w21 \\ w30 & w31 \end{bmatrix} = \begin{bmatrix} z0 & z1 \end{bmatrix}$$

$$\sigma(\begin{bmatrix} z0 & z1 \end{bmatrix}) = \begin{bmatrix} a0 & a1 \end{bmatrix}$$



Deep Neural Network

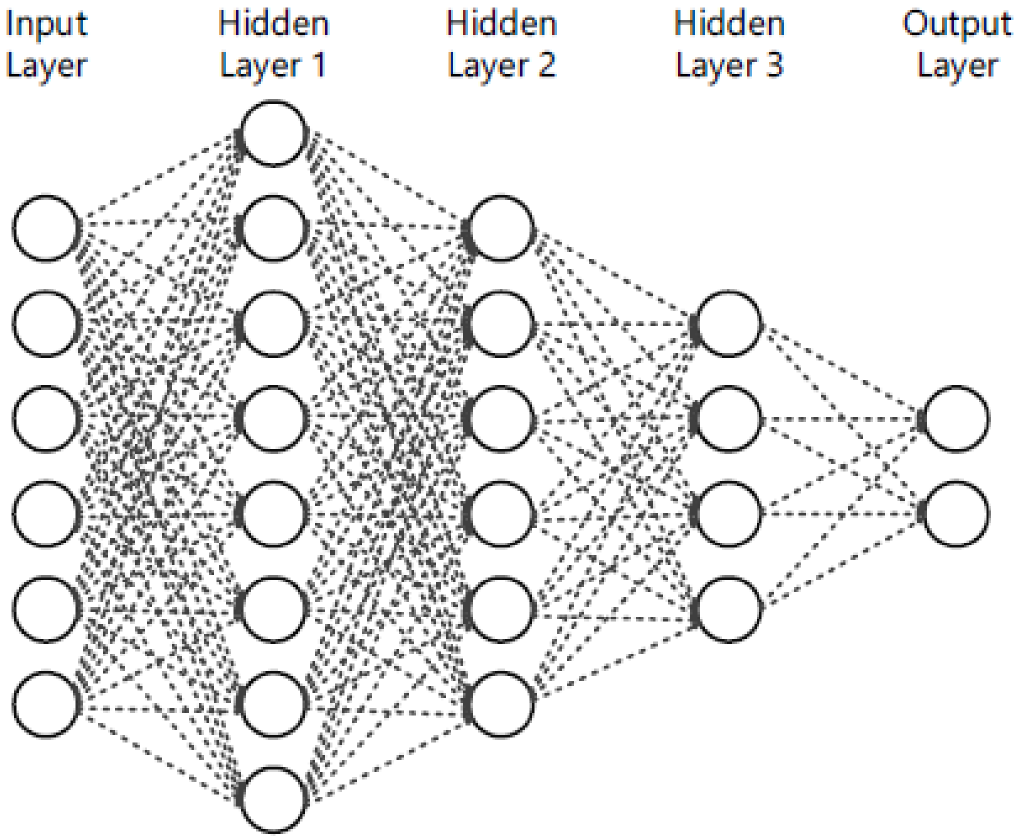
- ANN yang memiliki hidden layer lebih dari 2 disebut Deep Neural Network (DNN). Contoh model dari paper: Implementation of Systolic Co-processor for Deep Neural Network Inference based on SoC (https://ieeexplore.ieee.org/document/8649920).



Model ANN

Model MATLAB Deep Neural Network

- File model matlab: https://github.com/yohanes-erwin/pemrograman_zynq/blob/main/Aplikasi_ANN/model/model_ann.m



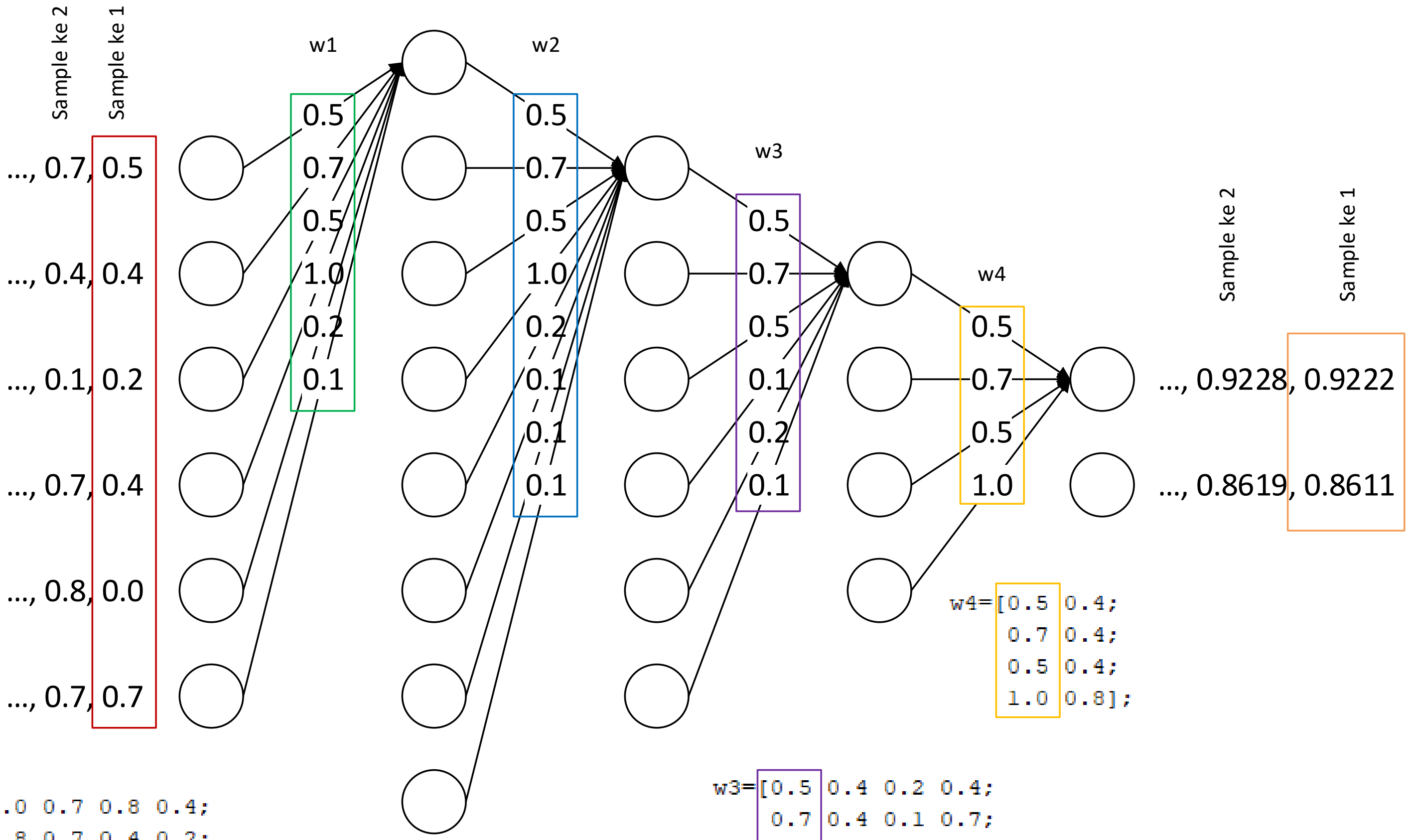
Ilustrasi Matrix pada Model

Node

Sample

```
x=[0.5 0.4 0.2 0.4 0.0 0.7;   Sample ke 1
   0.7 0.4 0.1 0.7 0.8 0.7;
   0.5 0.4 0.7 0.7 0.2 0.4;
   1.0 0.8 0.1 0.5 0.4 0.0;
   0.2 0.6 0.5 0.3 0.7 0.3;        ...
   0.1 0.8 0.5 0.1 0.4 0.4;
   0.1 0.9 0.9 0.6 0.7 0.3;
   0.1 0.1 0.5 0.3 0.4 0.2];  Sample ke 8
```

Node 1 ... Node 6

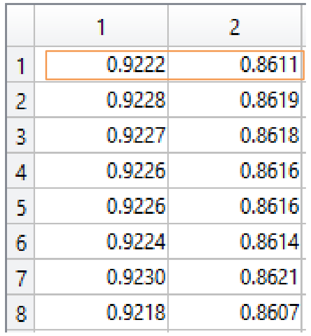


```
w4=[0.5 0.4;
    0.7 0.4;
    0.5 0.4;
    1.0 0.8];
```

```
w1=[0.5 0.4 0.2 0.4 0.0 0.7 0.8 0.4;
    0.7 0.4 0.1 0.7 0.8 0.7 0.4 0.2;
    0.5 0.4 0.7 0.7 0.2 0.4 0.9 0.8;
    1.0 0.8 0.1 0.5 0.4 0.0 0.4 0.9;
    0.2 0.6 0.5 0.3 0.7 0.3 0.8 0.3;
    0.1 0.8 0.5 0.1 0.4 0.4 0.4 0.7;];
```

```
w2=[0.5 0.4 0.2 0.4 0.0 0.7;
    0.7 0.4 0.1 0.7 0.8 0.7;
    0.5 0.4 0.7 0.7 0.2 0.4;
    1.0 0.8 0.1 0.5 0.4 0.0;
    0.2 0.6 0.5 0.3 0.7 0.3;
    0.1 0.8 0.5 0.1 0.4 0.4;
    0.1 0.9 0.9 0.6 0.7 0.3;
    0.1 0.1 0.5 0.3 0.4 0.2];
```

```
w3=[0.5 0.4 0.2 0.4;
    0.7 0.4 0.1 0.7;
    0.5 0.4 0.7 0.7;
    0.1 0.8 0.1 0.5;
    0.2 0.6 0.5 0.3;
    0.1 0.8 0.5 0.1];
```

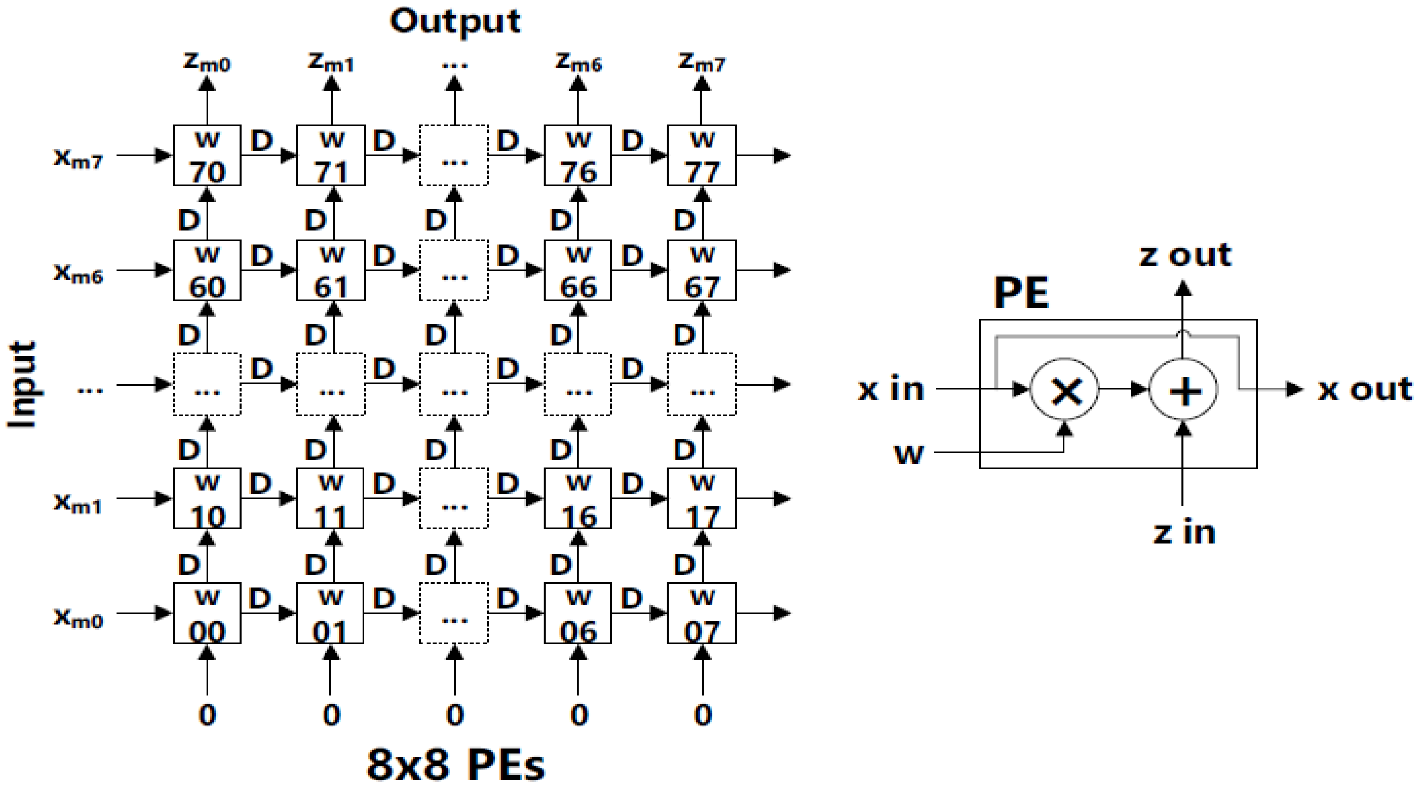


	1	2
1	0.9222	0.8611
2	0.9228	0.8619
3	0.9227	0.8618
4	0.9226	0.8616
5	0.9226	0.8616
6	0.9224	0.8614
7	0.9230	0.8621
8	0.9218	0.8607

Design Accelerator ANN

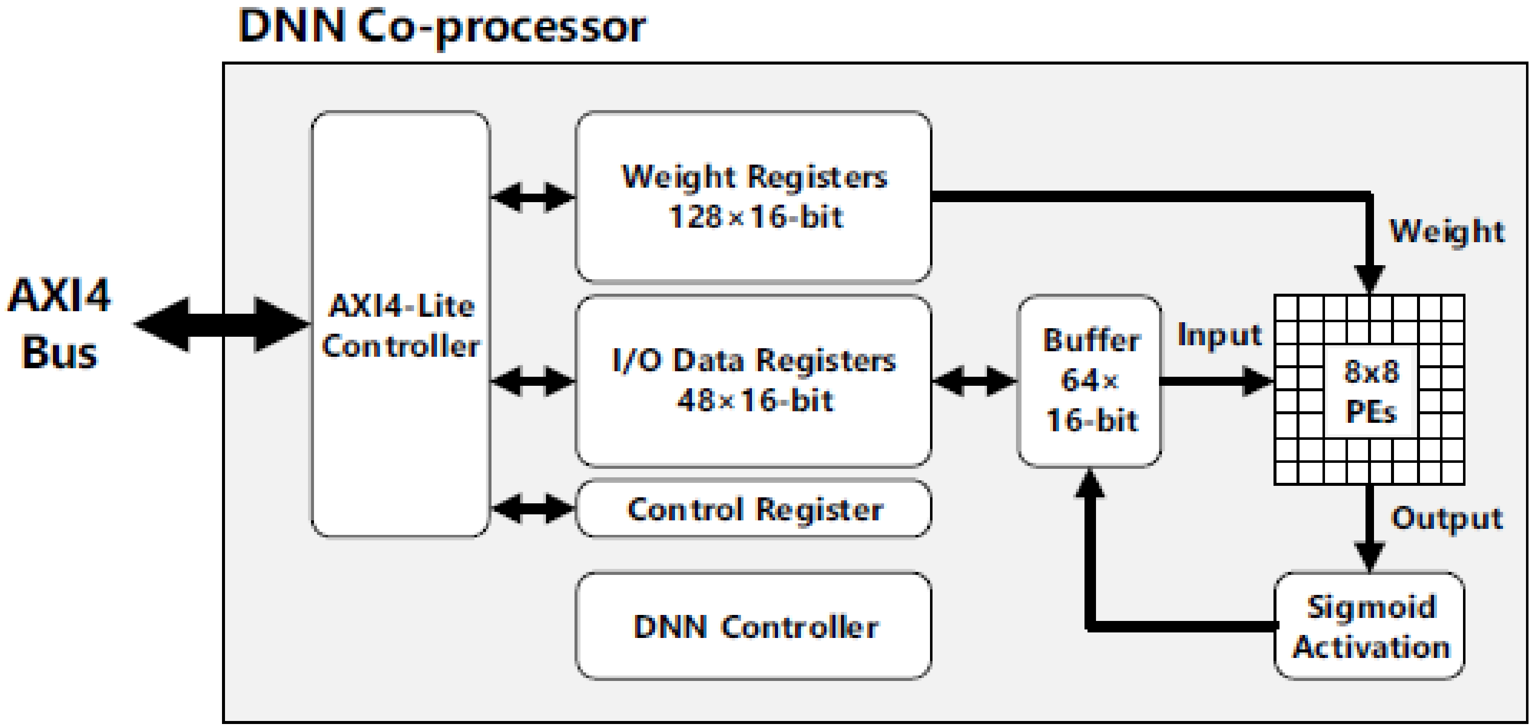
Perkalian Matrix

- Datapath systolic perkalian matrix untuk ukuran maximal perkalian 8x8.



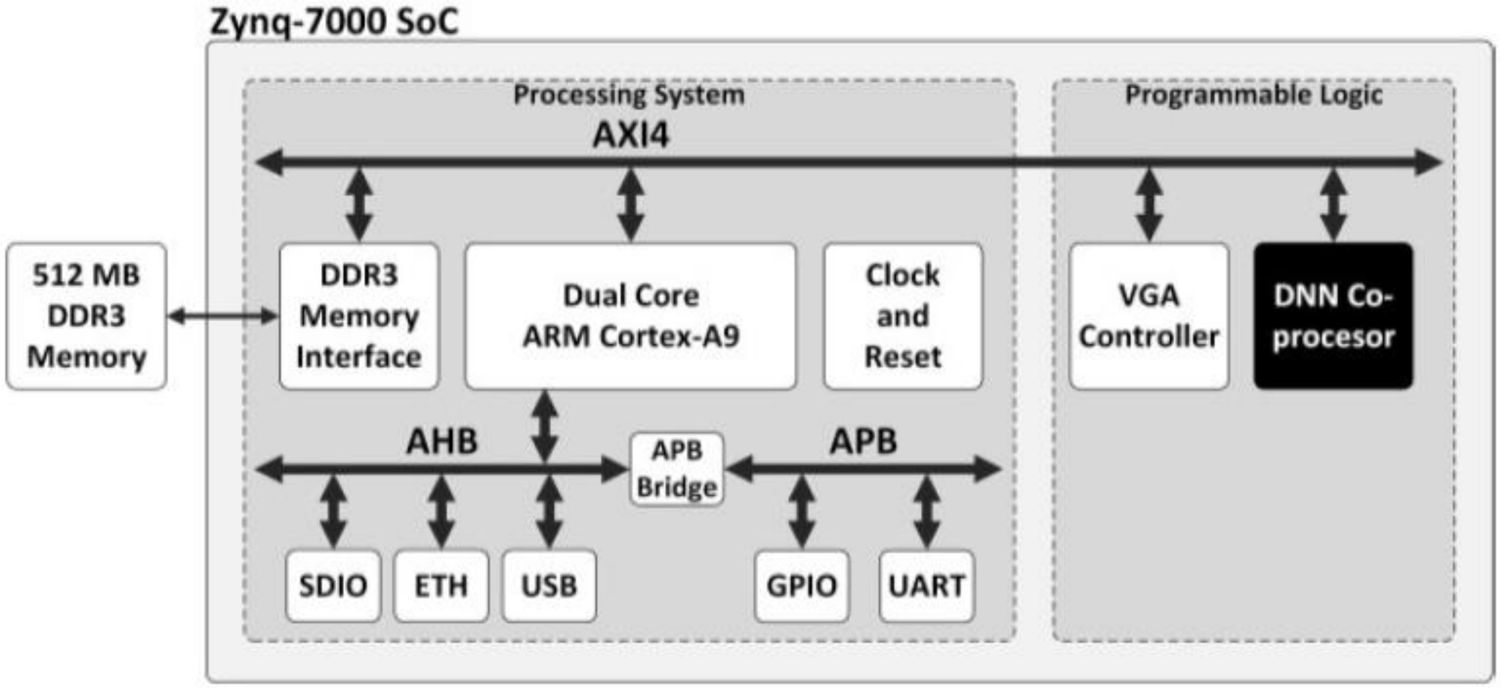
Design Accelerator DNN

- Design accelerator (co-processor) DNN dengan datapath systolic matrix 8x8, register untuk data, weight, dan interface AXI4-lite.



Design SoC DNN

- Design SoC secara keseluruhan terdiri dari DNN co-processor dan processing system tempat program C dijalankan untuk mengakses DNN co-processor tersebut.

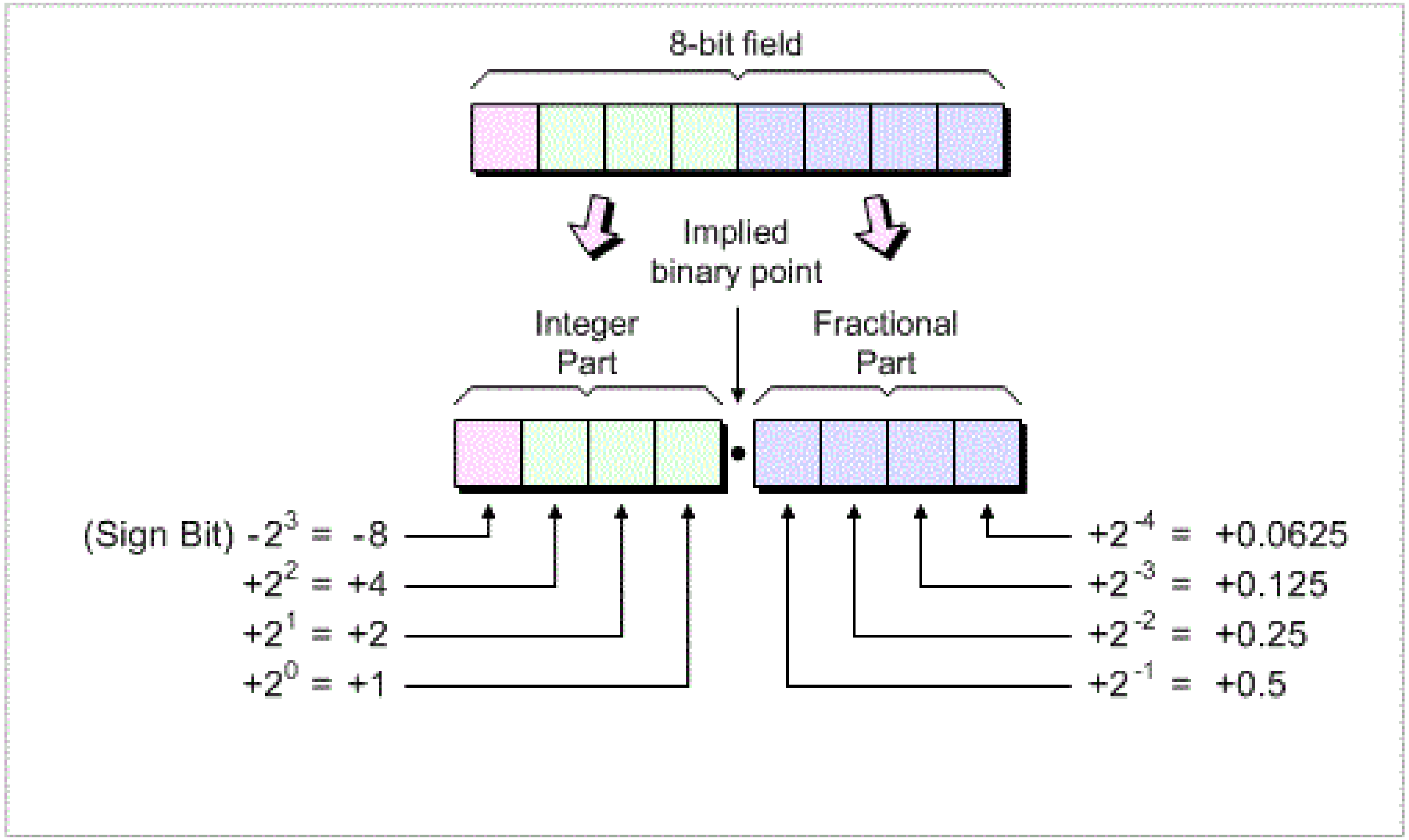


Kode Verilog Co-processor DNN

- File **module RTL**: https://github.com/yohanes-erwin/pemrograman_zynq/tree/main/Aplikasi_ANN/module
- File **testbench**: https://github.com/yohanes-erwin/pemrograman_zynq/tree/main/Aplikasi_ANN/testbench
- File **program C**: https://github.com/yohanes-erwin/pemrograman_zynq/tree/main/Aplikasi_ANN/program

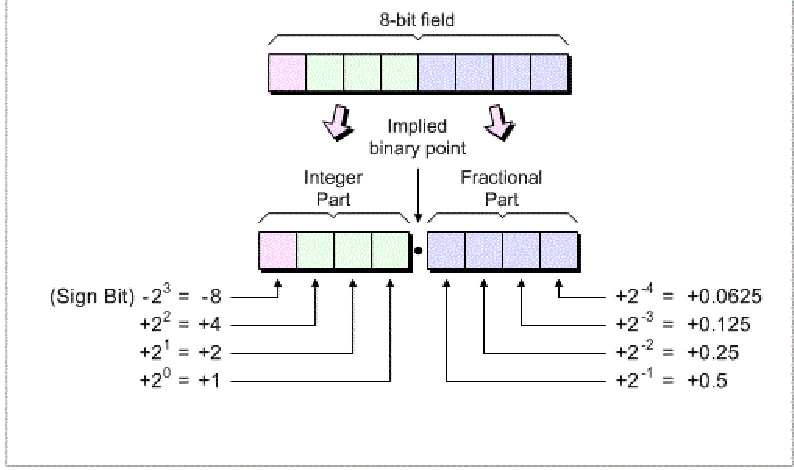
Bilangan Fixed Point

Fixed-Point



https://www.allaboutcircuits.com/technical-articles/fixed-point-representation-the-q-format-and-addition-examples/

Fixed-Point



- 01010110 = 86 (integer)
- 01010110 = 5.375 (fixed-point 1 sign 3 integer 4 fraction)

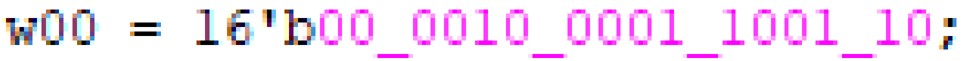
$$0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4} = 5.375$$

https://www.allaboutcircuits.com/technical-articles/fixed-point-representation-the-q-format-and-addition-examples/

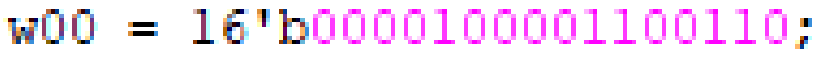
Fixed-Point

- Fixed-Point yang digunakan di perkalian marix dan ANN adalah 16-bit dengan
  - 1 sign bit (0: positif, 1: negative)
  - 5 integer bit
  - 10 fraction bit

Fixed-Point

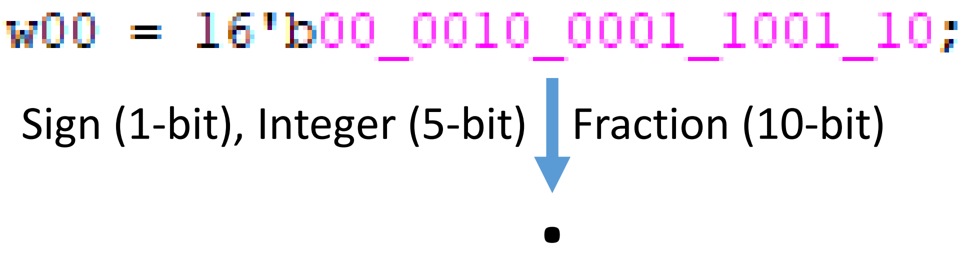


```
w00 = 16'b00_0010_0001_1001_10;
```



```
w00 = 16'b0000100001100110;
```

Tanda _ (underscore) hanya untuk memudahkan pembacaan bit tidak ada pengaruhnya dengan nilainya. Kedua variable tersebut akan bernilai sama. Compiler akan mengabaikan tanda _.

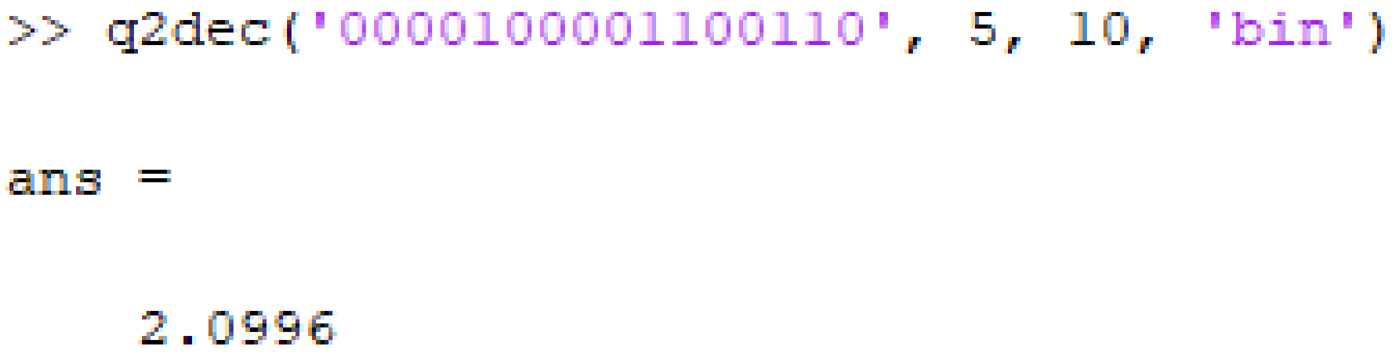


```
w00 = 16'b00_0010_0001_1001_10;
```

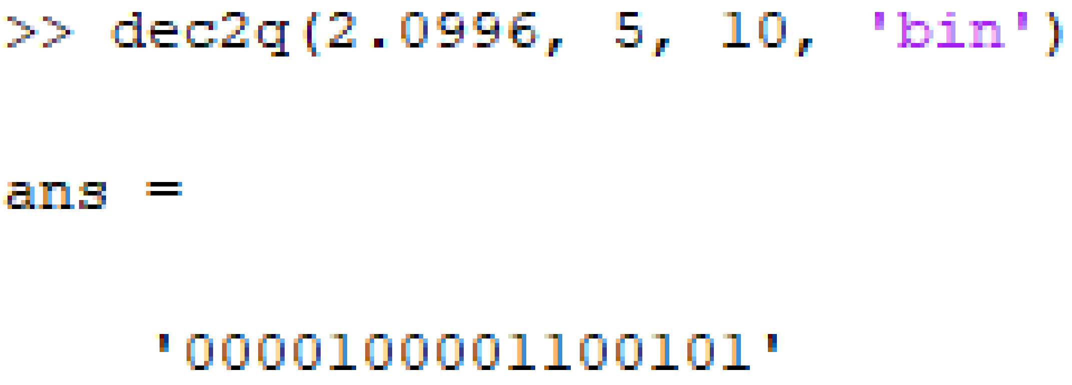
Sign (1-bit), Integer (5-bit) Fraction (10-bit)

.

Fixed-Point



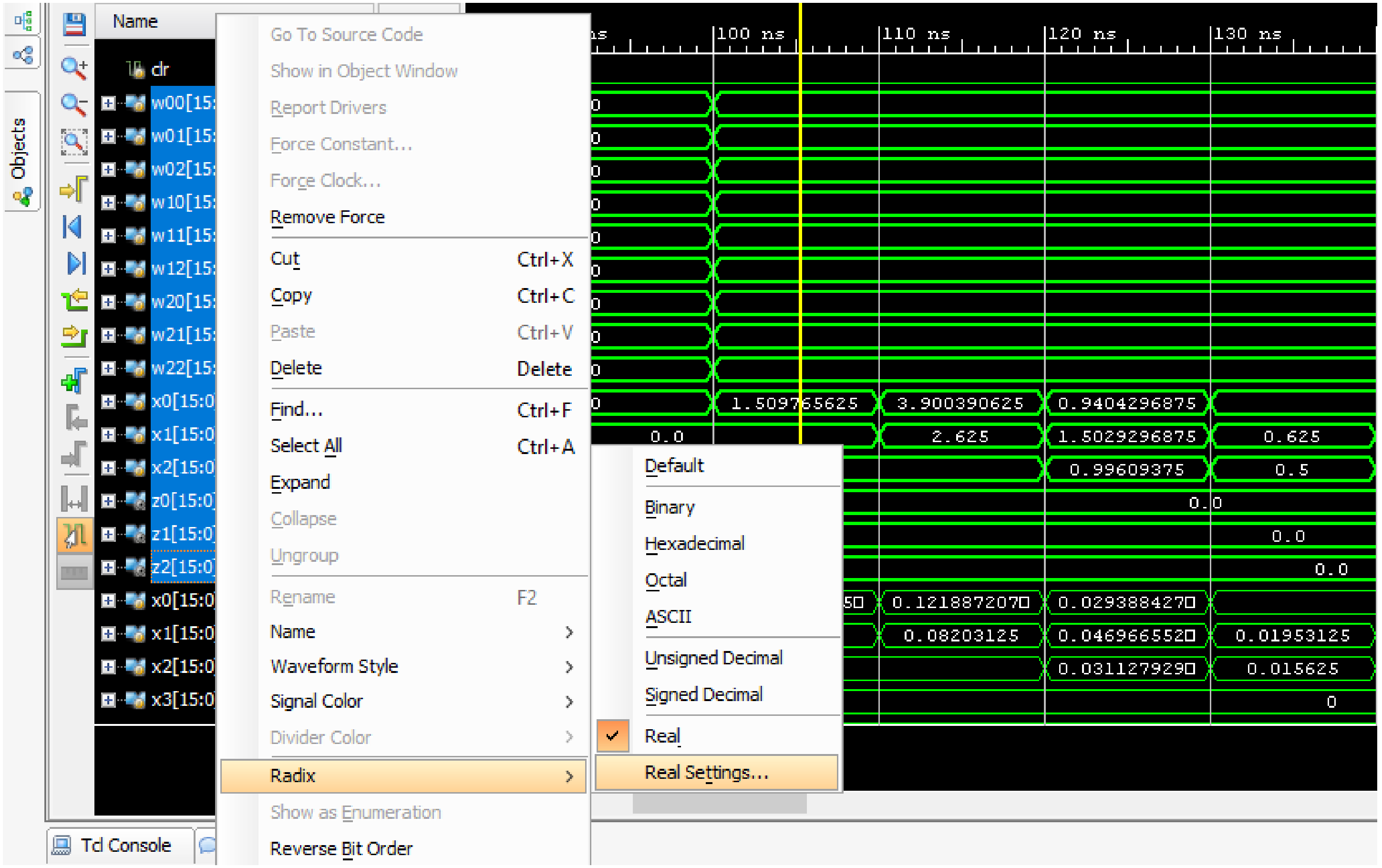
```
>> q2dec('0000100001100110', 5, 10, 'bin')

ans =

    2.0996
```

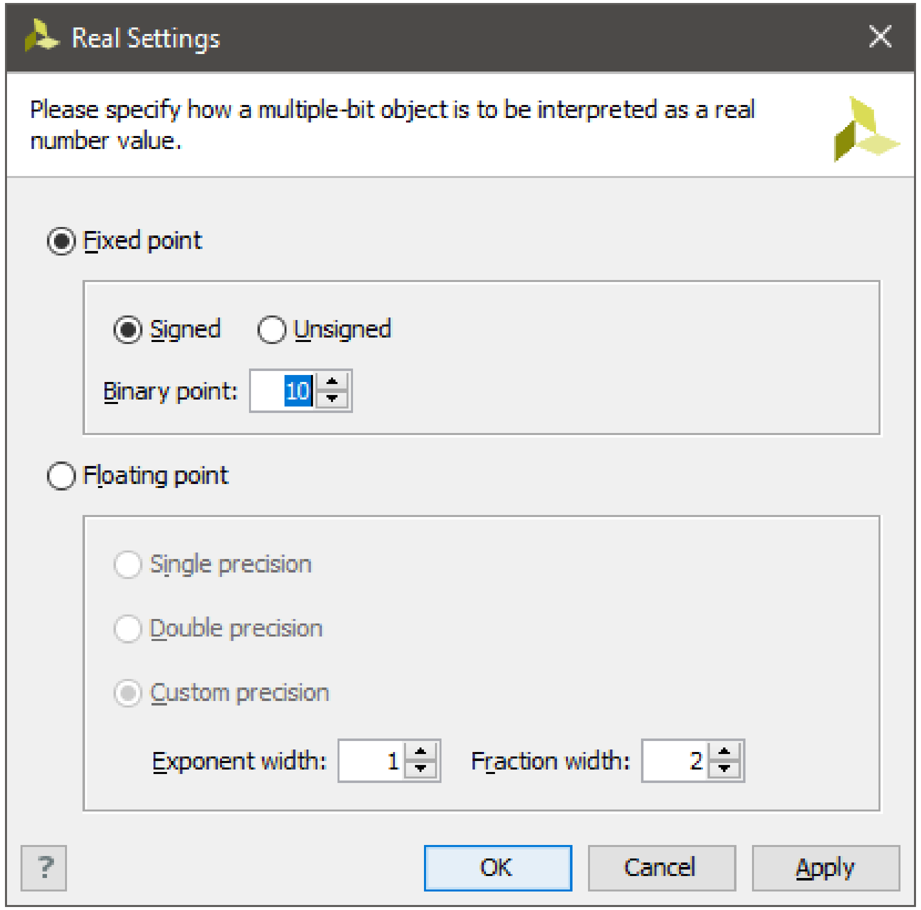


```
>> dec2q(2.0996, 5, 10, 'bin')

ans =

    '0000100001100101'
```

https://www.mathworks.com/matlabcentral/fileexchange/61670-fixed-point-q-format-to-decimal-converter
https://www.mathworks.com/matlabcentral/fileexchange/61669-decimal-to-fixed-point-q-format-converter

Cara Mengganti Tampilan ke Fixed Point




Real Settings

Please specify how a multiple-bit object is to be interpreted as a real number value.

● Fixed point

● Signed ○ Unsigned

Binary point: 10

○ Floating point

○ Single precision

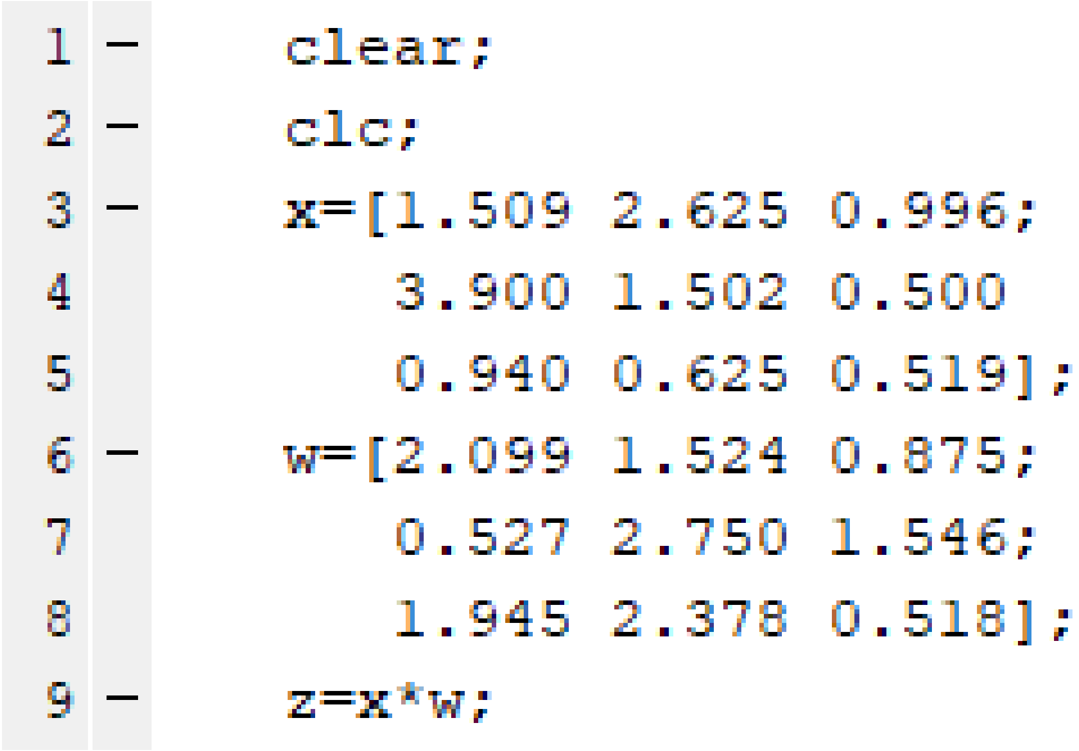
○ Double precision

○ Custom precision

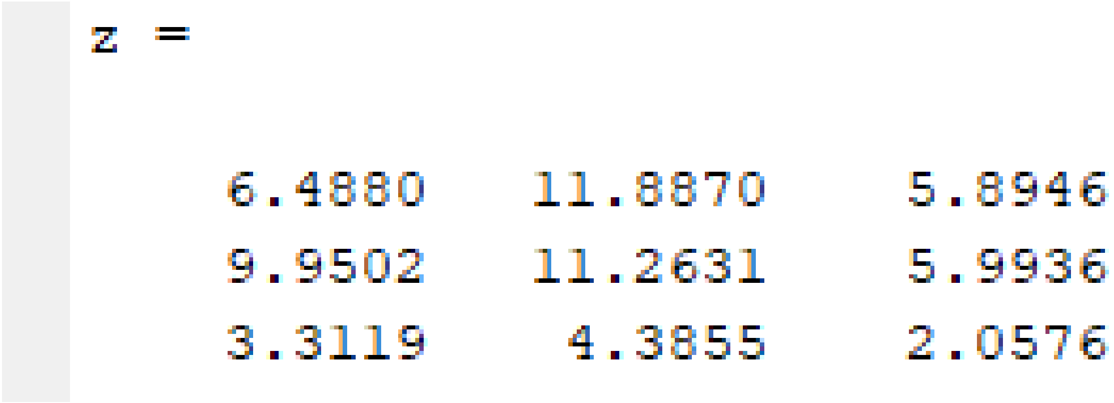
Exponent width: 1 Fraction width: 2

Simulasi Perkalian Matrix

Perkalian Matrix

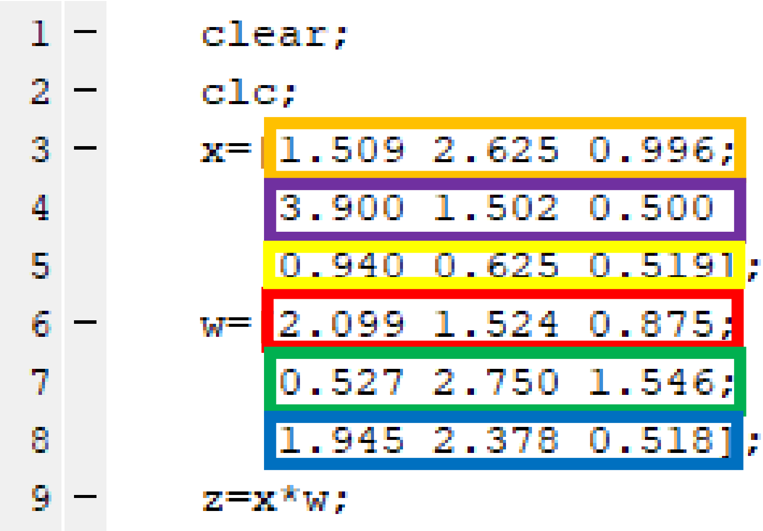


```
clear;
clc;
x=[1.509 2.625 0.996;
   3.900 1.502 0.500
   0.940 0.625 0.519];
w=[2.099 1.524 0.875;
   0.527 2.750 1.546;
   1.945 2.378 0.518];
z=x*w;
```

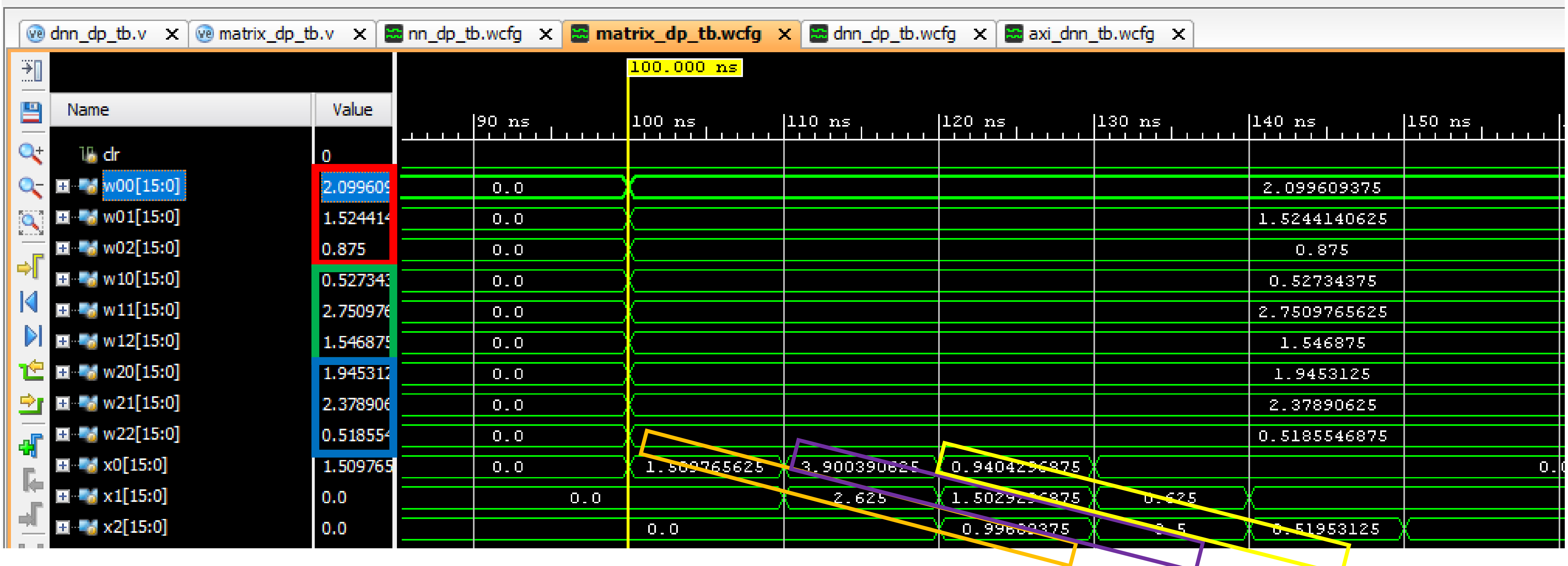


```
z =

    6.4880   11.8870    5.8946
    9.9502   11.2631    5.9936
    3.3119    4.3855    2.0576
```

Input

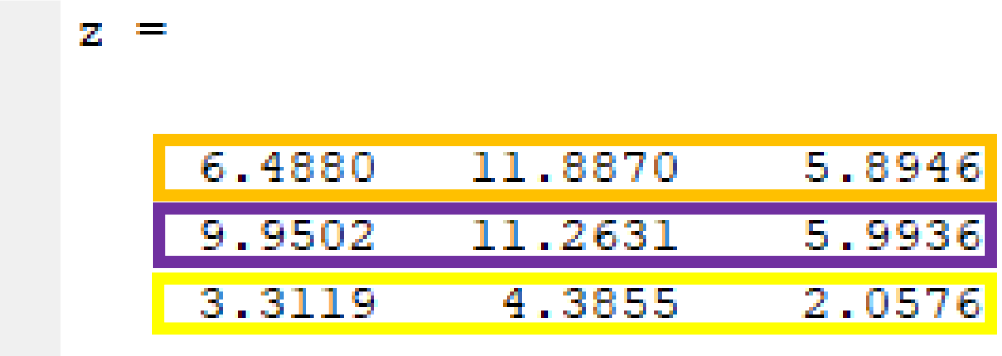


```
clear;
clc;
x=[1.509 2.625 0.996;
   3.900 1.502 0.500
   0.940 0.625 0.519];
w=[2.099 1.524 0.875;
   0.527 2.750 1.546;
   1.945 2.378 0.518];
z=x*w;
```

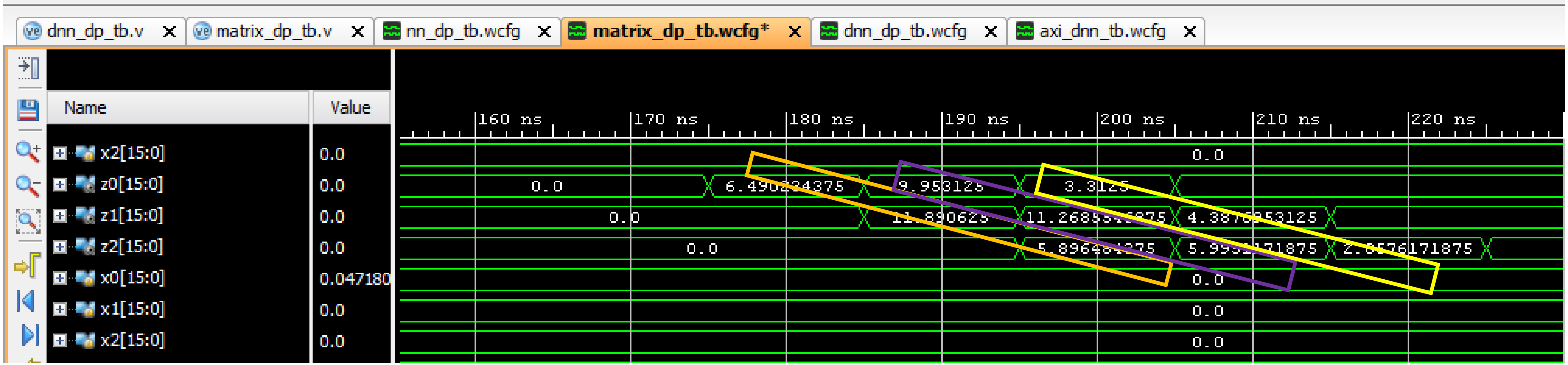


Name	Value
dr	0
w00[15:0]	2.099609
w01[15:0]	1.524414
w02[15:0]	0.875
w10[15:0]	0.527343
w11[15:0]	2.750976
w12[15:0]	1.546875
w20[15:0]	1.945312
w21[15:0]	2.378906
w22[15:0]	0.518554
x0[15:0]	1.509765
x1[15:0]	0.0
x2[15:0]	0.0

Output



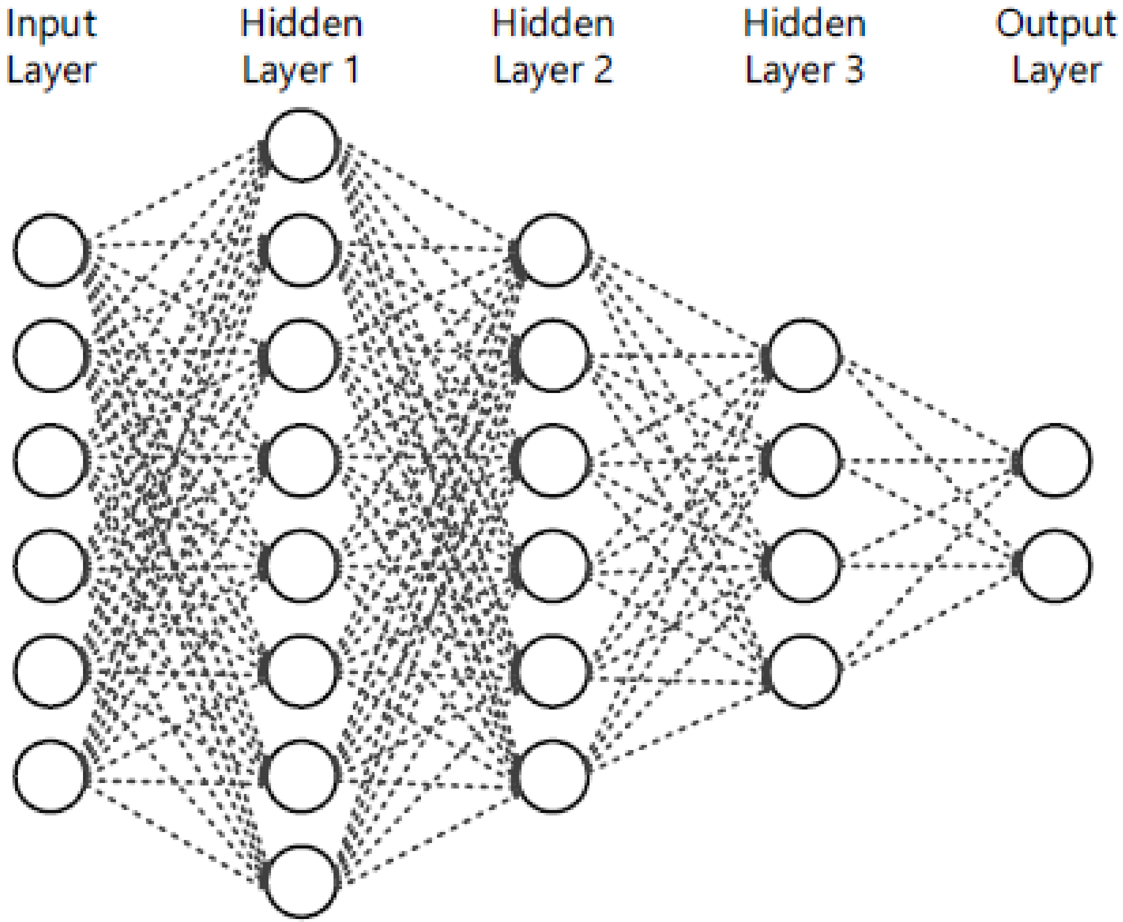
```
z =

    6.4880   11.8870    5.8946
    9.9502   11.2631    5.9936
    3.3119    4.3855    2.0576
```



Name	Value	Waveform
x2[15:0]	0.0	0.0
z0[15:0]	0.0	0.0 · 6.490234375 · 9.953125 · 3.3125
z1[15:0]	0.0	0.0 · 11.890625 · 11.2685546875 · 4.3876953125
z2[15:0]	0.0	0.0 · 5.896484375 · 5.9951171875 · 2.0576171875
x0[15:0]	0.047180	0.0
x1[15:0]	0.0	0.0
x2[15:0]	0.0	0.0

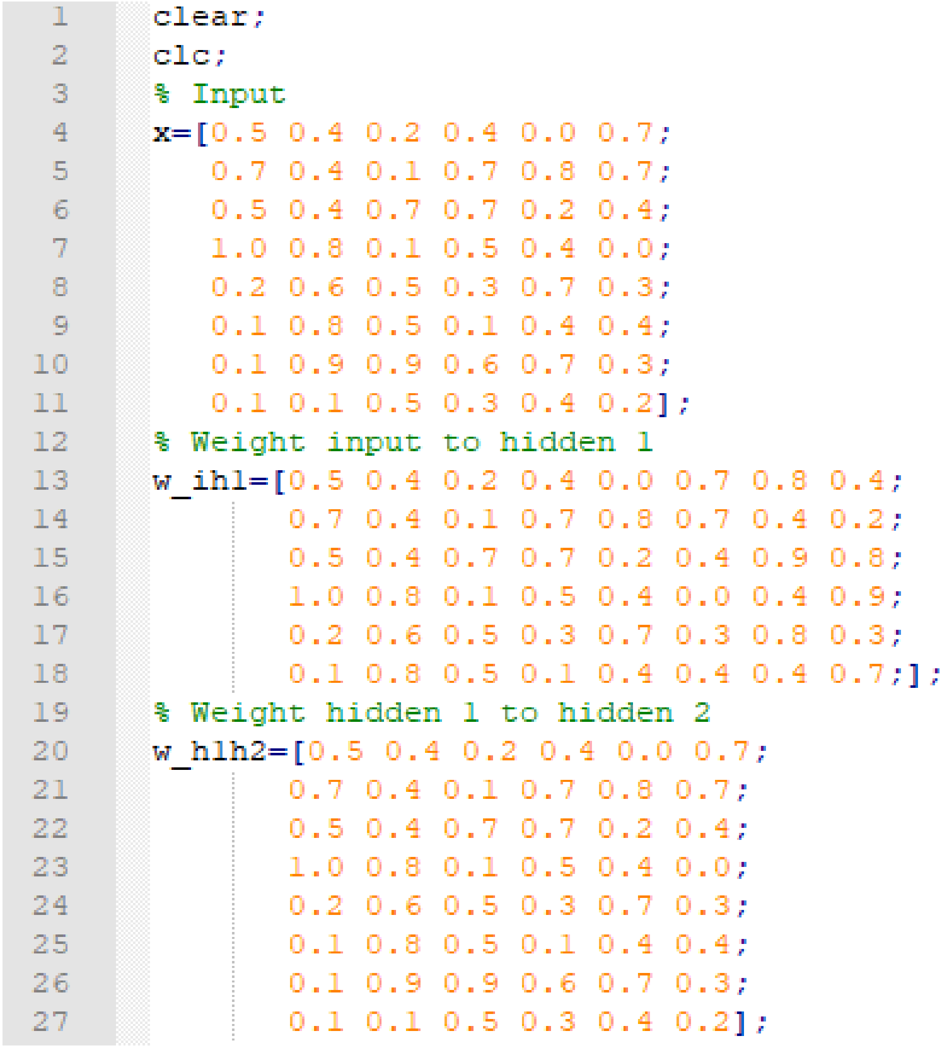
Simulasi Proses 1 Layer ANN

Model ANN

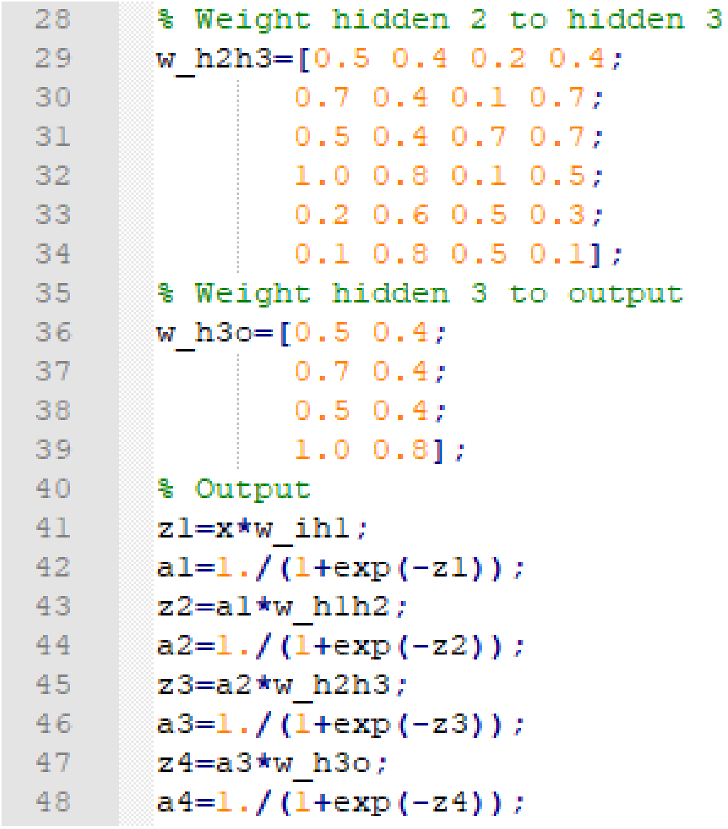


Implementation of Systolic Co-processor for Deep Neural Network Inference based on SoC

Model ANN di Matlab



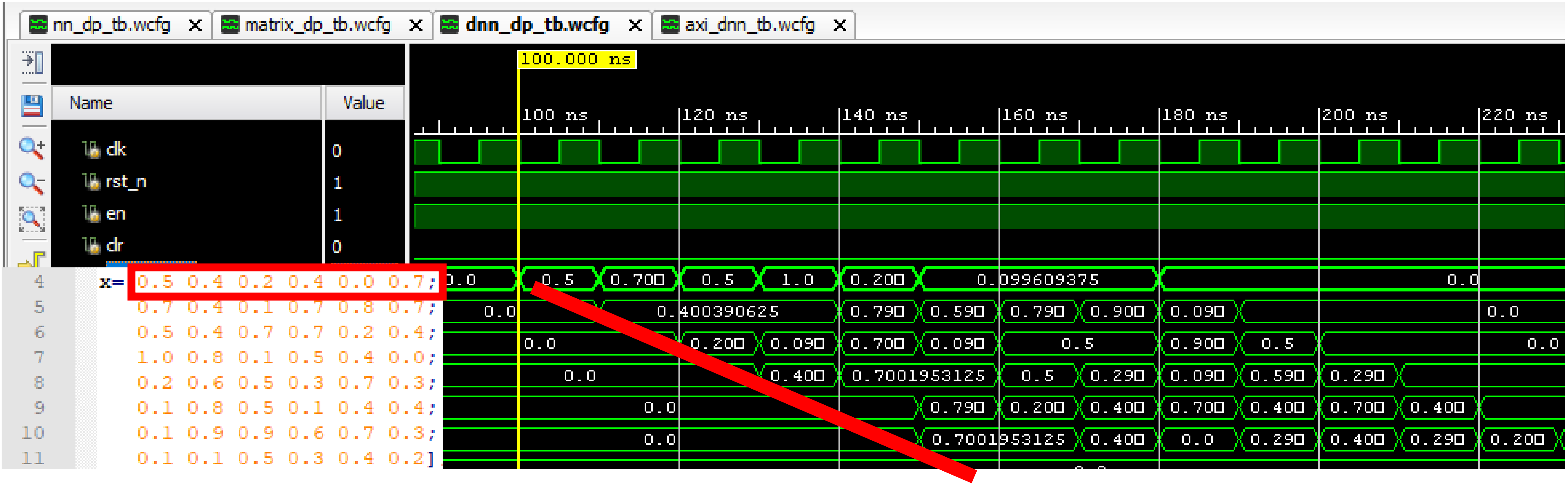
```
clear;
clc;
% Input
x=[0.5 0.4 0.2 0.4 0.0 0.7;
   0.7 0.4 0.1 0.7 0.8 0.7;
   0.5 0.4 0.7 0.7 0.2 0.4;
   1.0 0.8 0.1 0.5 0.4 0.0;
   0.2 0.6 0.5 0.3 0.7 0.3;
   0.1 0.8 0.5 0.1 0.4 0.4;
   0.1 0.9 0.9 0.6 0.7 0.3;
   0.1 0.1 0.5 0.3 0.4 0.2];
% Weight input to hidden 1
w_ih1=[0.5 0.4 0.2 0.4 0.0 0.7 0.8 0.4;
       0.7 0.4 0.1 0.7 0.8 0.7 0.4 0.2;
       0.5 0.4 0.7 0.7 0.2 0.4 0.9 0.8;
       1.0 0.8 0.1 0.5 0.4 0.0 0.4 0.9;
       0.2 0.6 0.5 0.3 0.7 0.3 0.8 0.3;
       0.1 0.8 0.5 0.1 0.4 0.4 0.4 0.7;];
% Weight hidden 1 to hidden 2
w_h1h2=[0.5 0.4 0.2 0.4 0.0 0.7;
       0.7 0.4 0.1 0.7 0.8 0.7;
       0.5 0.4 0.7 0.7 0.2 0.4;
       1.0 0.8 0.1 0.5 0.4 0.0;
       0.2 0.6 0.5 0.3 0.7 0.3;
       0.1 0.8 0.5 0.1 0.4 0.4;
       0.1 0.9 0.9 0.6 0.7 0.3;
       0.1 0.1 0.5 0.3 0.4 0.2];
```



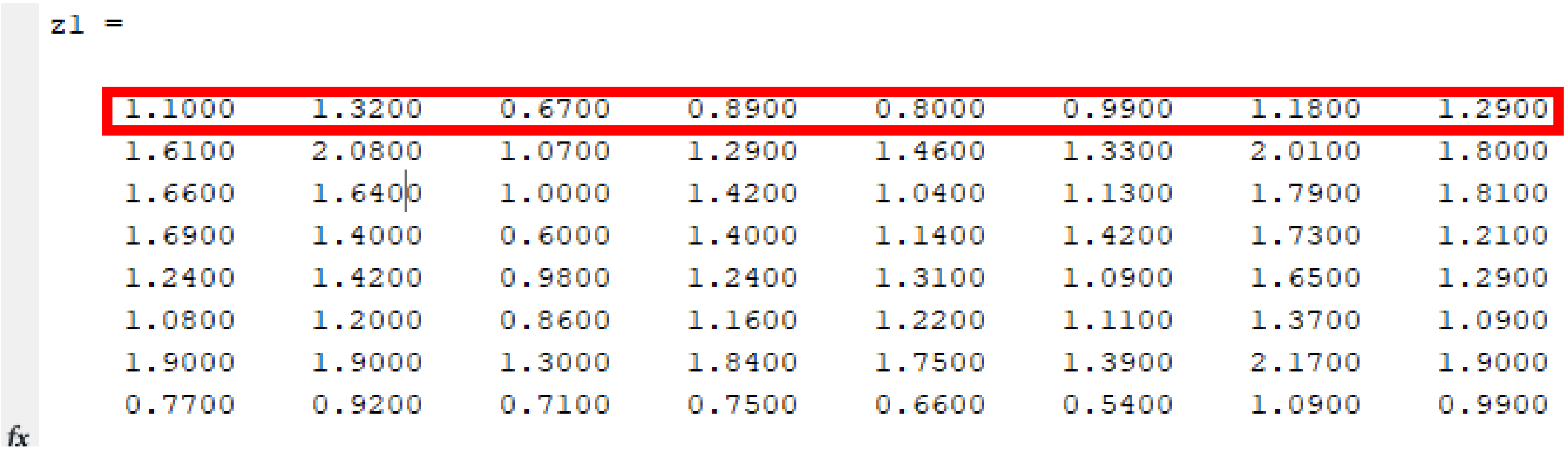
```
% Weight hidden 2 to hidden 3
w_h2h3=[0.5 0.4 0.2 0.4;
       0.7 0.4 0.1 0.7;
       0.5 0.4 0.7 0.7;
       1.0 0.8 0.1 0.5;
       0.2 0.6 0.5 0.3;
       0.1 0.8 0.5 0.1];
% Weight hidden 3 to output
w_h3o=[0.5 0.4;
       0.7 0.4;
       0.5 0.4;
       1.0 0.8];
% Output
z1=x*w_ih1;
a1=1./(1+exp(-z1));
z2=a1*w_h1h2;
a2=1./(1+exp(-z2));
z3=a2*w_h2h3;
a3=1./(1+exp(-z3));
z4=a3*w_h3o;
a4=1./(1+exp(-z4));
```

Simulasi 1 Layer

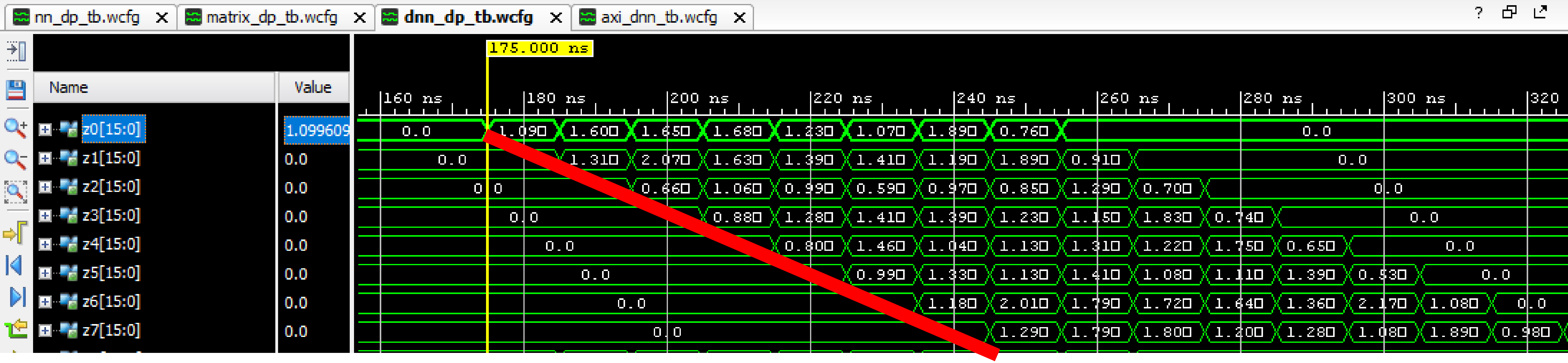
- dnn_dp_tb.v mensimulasikan satu layer pertama



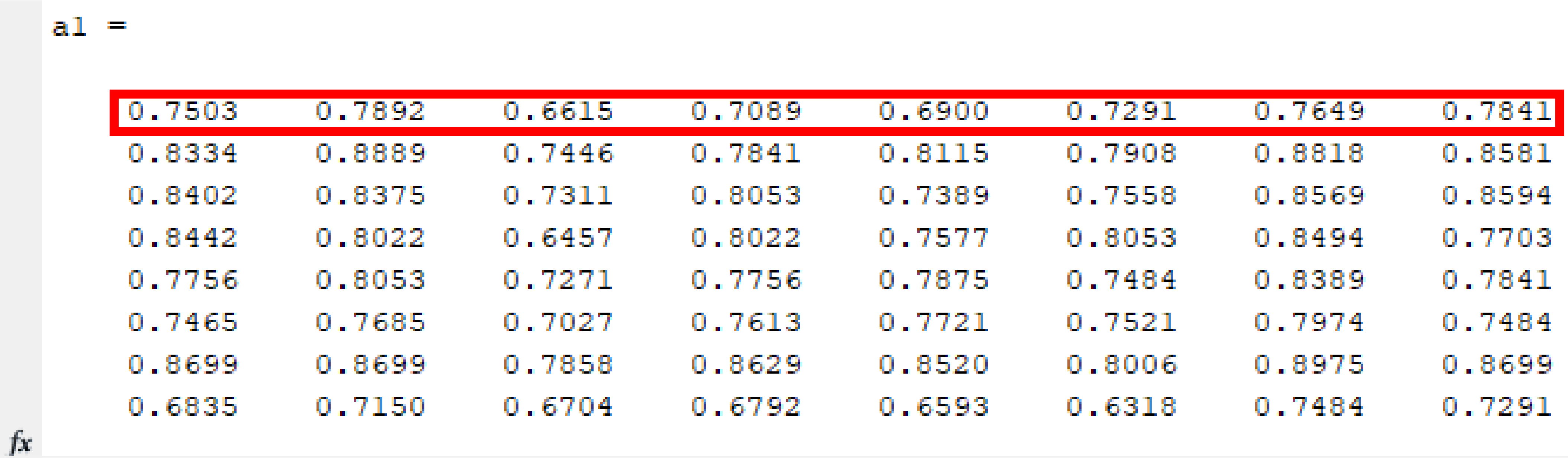
```
x= 0.5 0.4 0.2 0.4 0.0 0.7;
   0.7 0.4 0.1 0.7 0.8 0.7;
   0.5 0.4 0.7 0.7 0.2 0.4;
   1.0 0.8 0.1 0.5 0.4 0.0;
   0.2 0.6 0.5 0.3 0.7 0.3;
   0.1 0.8 0.5 0.1 0.4 0.4;
   0.1 0.9 0.9 0.6 0.7 0.3;
   0.1 0.1 0.5 0.3 0.4 0.2]
```



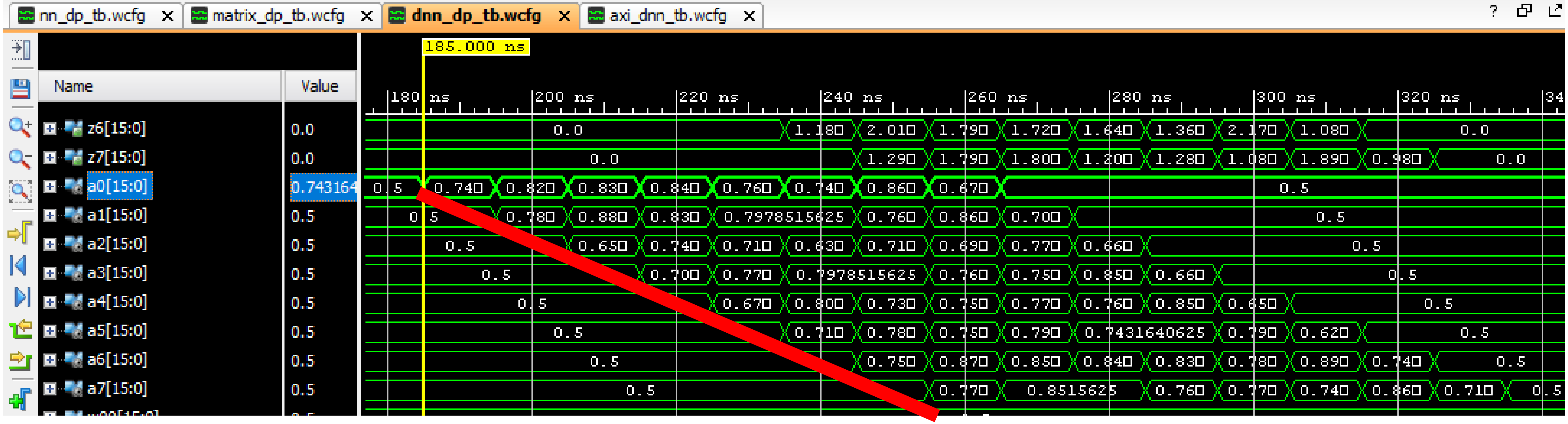
```
z1 =

    1.1000    1.3200    0.6700    0.8900    0.8000    0.9900    1.1800    1.2900
    1.6100    2.0800    1.0700    1.2900    1.4600    1.3300    2.0100    1.8000
    1.6600    1.6400    1.0000    1.4200    1.0400    1.1300    1.7900    1.8100
    1.6900    1.4000    0.6000    1.4000    1.1400    1.4200    1.7300    1.2100
    1.2400    1.4200    0.9800    1.2400    1.3100    1.0900    1.6500    1.2900
    1.0800    1.2000    0.8600    1.1600    1.2200    1.1100    1.3700    1.0900
    1.9000    1.9000    1.3000    1.8400    1.7500    1.3900    2.1700    1.9000
    0.7700    0.9200    0.7100    0.7500    0.6600    0.5400    1.0900    0.9900
```



Name	Value
z0[15:0]	1.099609
z1[15:0]	0.0
z2[15:0]	0.0
z3[15:0]	0.0
z4[15:0]	0.0
z5[15:0]	0.0
z6[15:0]	0.0
z7[15:0]	0.0



```
a1 =

    0.7503    0.7892    0.6615    0.7089    0.6900    0.7291    0.7649    0.7841
    0.8334    0.8889    0.7446    0.7841    0.8115    0.7908    0.8818    0.8581
    0.8402    0.8375    0.7311    0.8053    0.7389    0.7558    0.8569    0.8594
    0.8442    0.8022    0.6457    0.8022    0.7577    0.8053    0.8494    0.7703
    0.7756    0.8053    0.7271    0.7756    0.7875    0.7484    0.8389    0.7841
    0.7465    0.7685    0.7027    0.7613    0.7721    0.7521    0.7974    0.7484
    0.8699    0.8699    0.7858    0.8629    0.8520    0.8006    0.8975    0.8699
    0.6835    0.7150    0.6704    0.6792    0.6593    0.6318    0.7484    0.7291
```

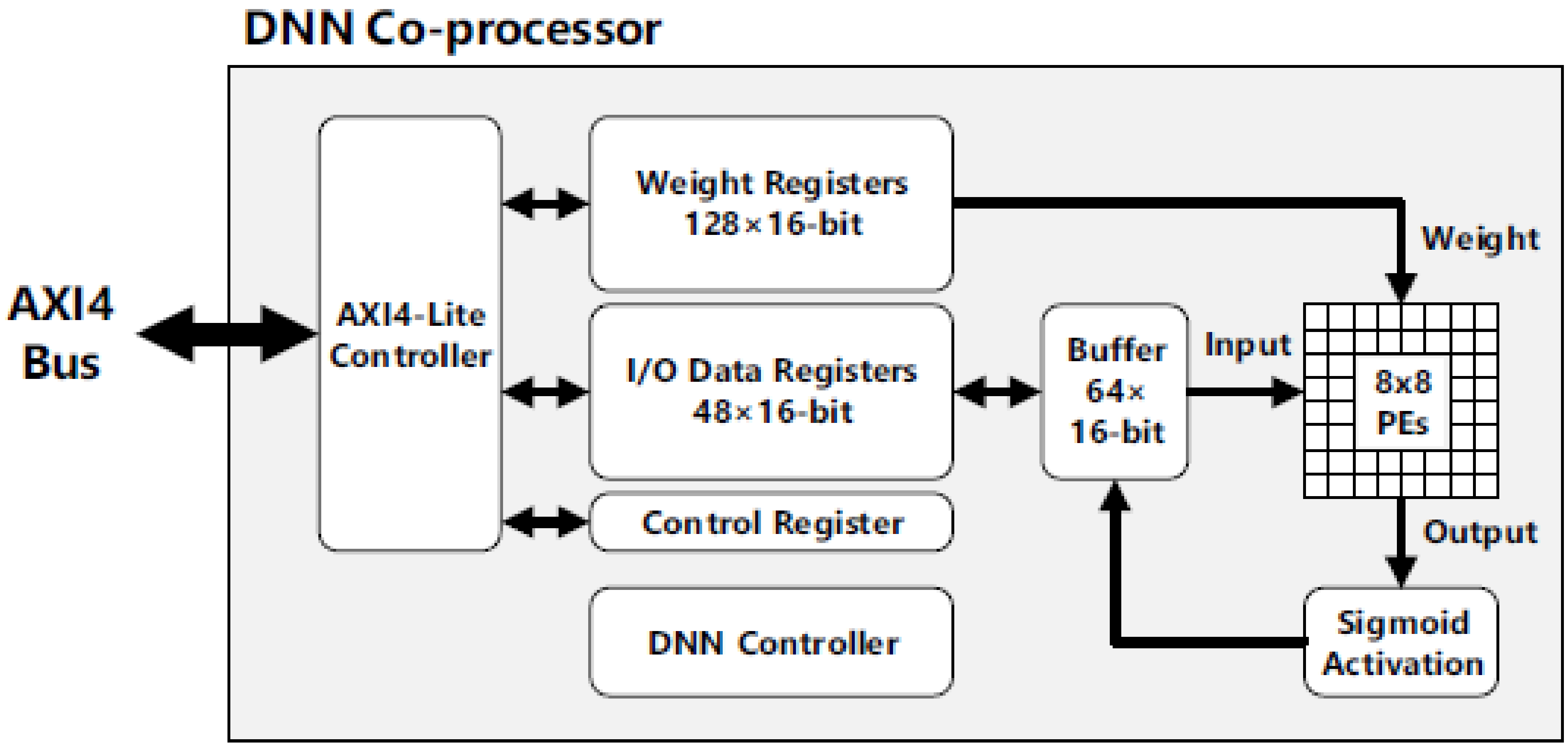


Name	Value
z6[15:0]	0.0
z7[15:0]	0.0
a0[15:0]	0.743164
a1[15:0]	0.5
a2[15:0]	0.5
a3[15:0]	0.5
a4[15:0]	0.5
a5[15:0]	0.5
a6[15:0]	0.5
a7[15:0]	0.5
w00[15:0]	0.5

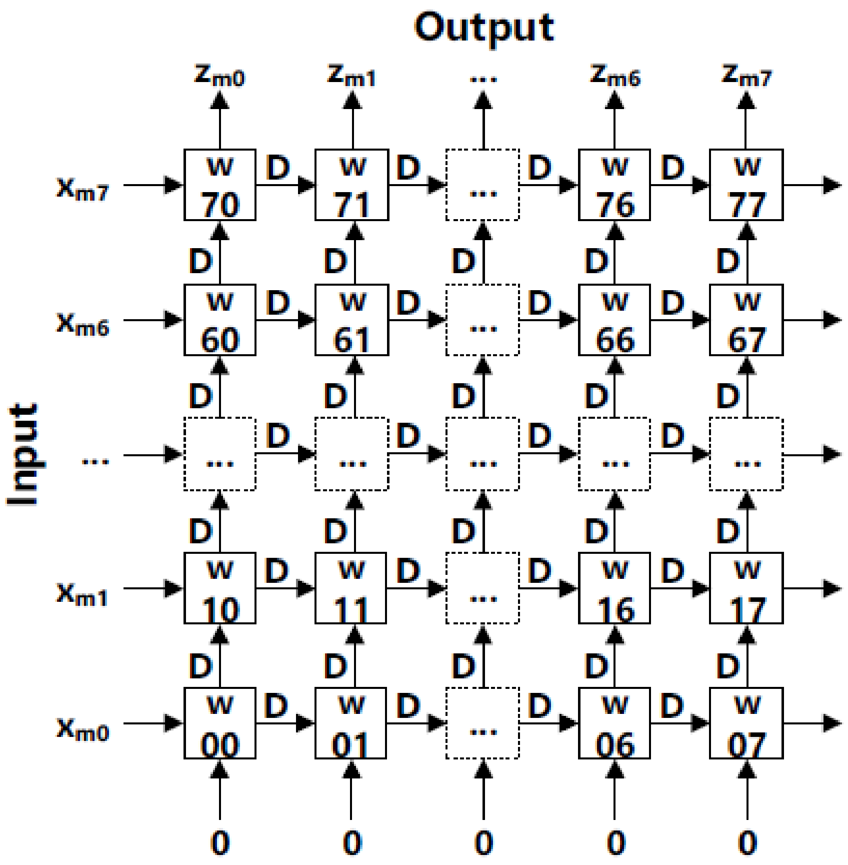
```
z6   0.0  1.18□  2.01□  1.79□  1.72□  1.64□  1.36□  2.17□  1.08□  0.0
z7   0.0  1.29□  1.79□  1.80□  1.20□  1.28□  1.08□  1.89□  0.98□  0.0
a0   0.5  0.74□  0.82□  0.83□  0.84□  0.76□  0.74□  0.86□  0.67□  0.5
a1   0.5  0.78□  0.88□  0.83□  0.7978515625  0.76□  0.86□  0.70□  0.5
a2   0.5  0.65□  0.74□  0.71□  0.63□  0.71□  0.69□  0.77□  0.66□  0.5
a3   0.5  0.70□  0.77□  0.7978515625  0.76□  0.75□  0.85□  0.66□  0.5
a4   0.5  0.67□  0.80□  0.73□  0.75□  0.77□  0.76□  0.85□  0.65□  0.5
a5   0.5  0.71□  0.78□  0.75□  0.79□  0.7431640625  0.79□  0.62□  0.5
a6   0.5  0.75□  0.87□  0.85□  0.84□  0.83□  0.78□  0.89□  0.74□  0.5
a7   0.5  0.77□  0.8515625  0.76□  0.77□  0.74□  0.86□  0.71□  0.5
```

Simulasi Proses Lebih dari 1 Layer

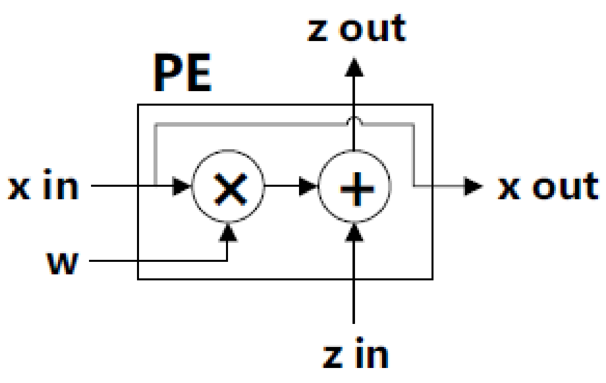
Block Diagram



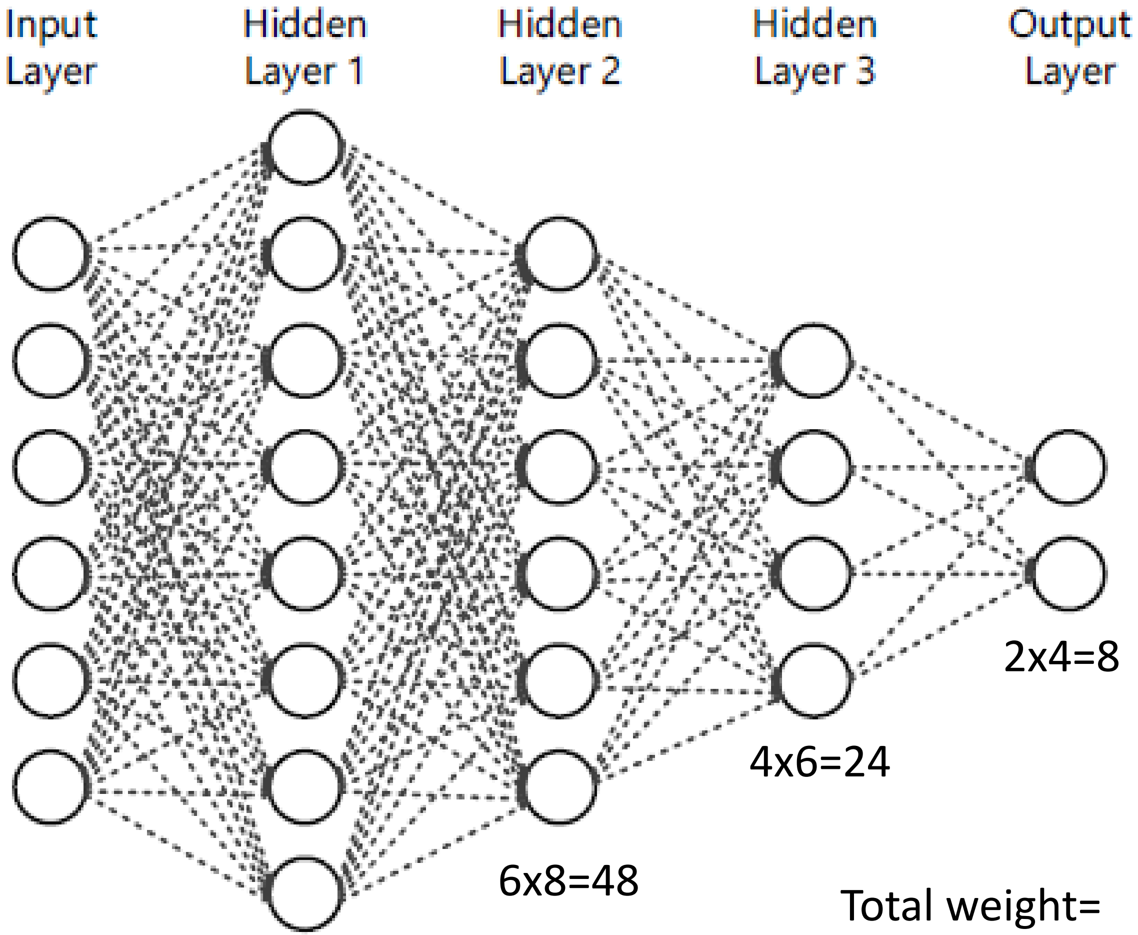
Processor Perkalian Matrix



8x8 PEs



Jumlah Register Weight



2x4=8

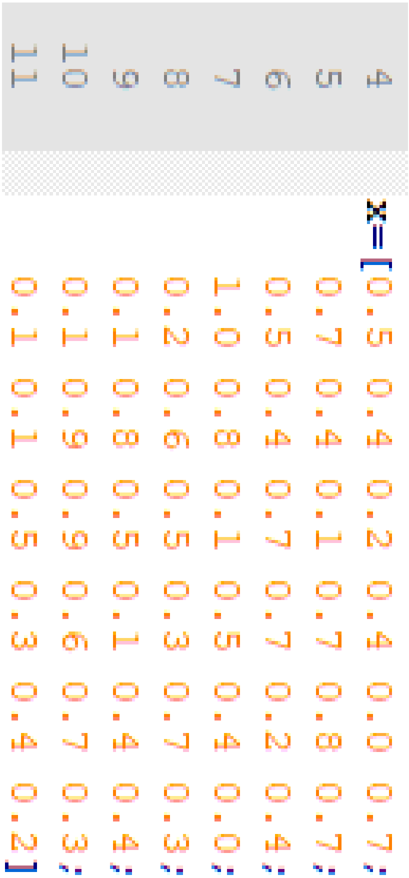
4x6=24

6x8=48

Total weight=
48+48+24+8=128

8x6=48

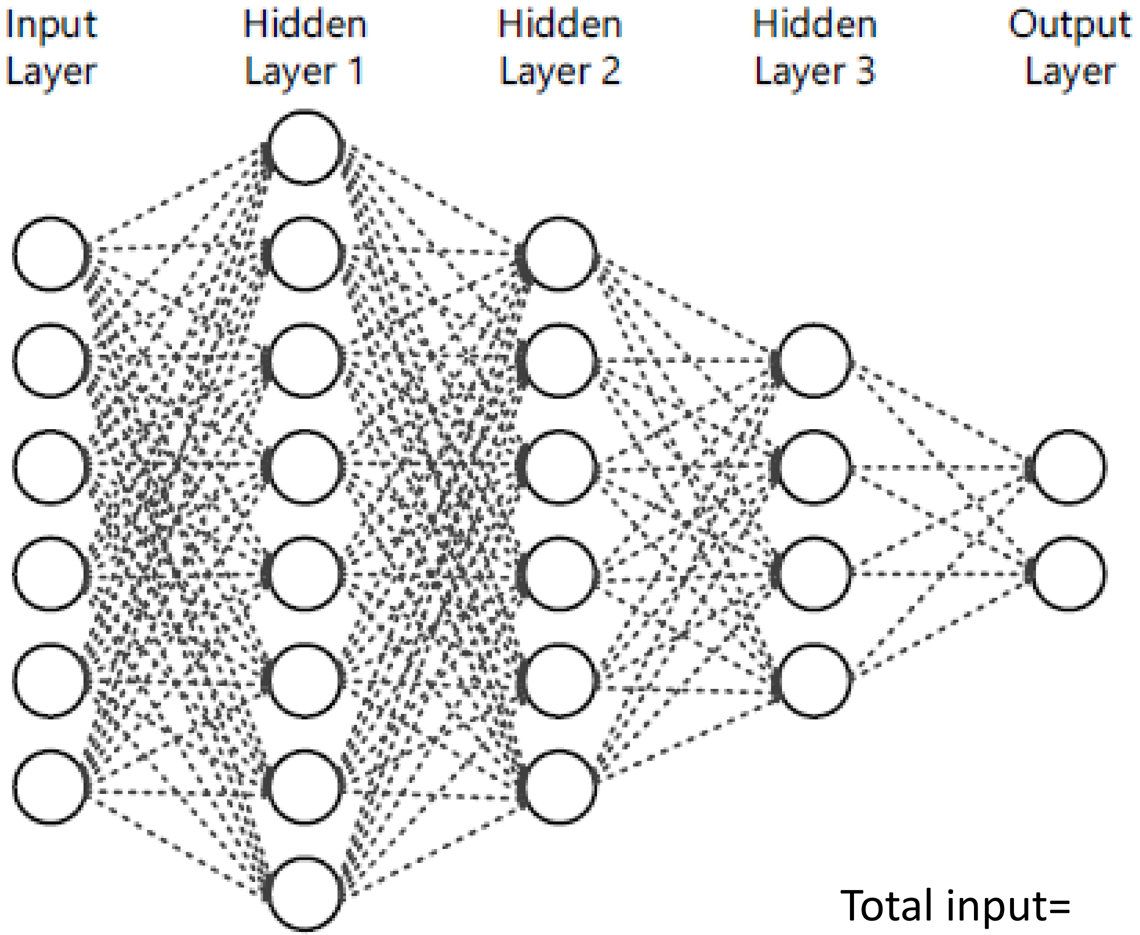
Jumlah Register Input



```
x=[0.5 0.4 0.2 0.4 0.0 0.7;
0.7 0.4 0.1 0.7 0.8 0.7;
0.5 0.4 0.7 0.7 0.2 0.4;
1.0 0.8 0.1 0.5 0.4 0.0;
0.2 0.6 0.5 0.3 0.7 0.3;
0.1 0.8 0.5 0.1 0.4 0.4;
0.1 0.9 0.9 0.6 0.7 0.3;
0.1 0.1 0.5 0.3 0.4 0.2]
```

Sample ke-8

Sample ke-1



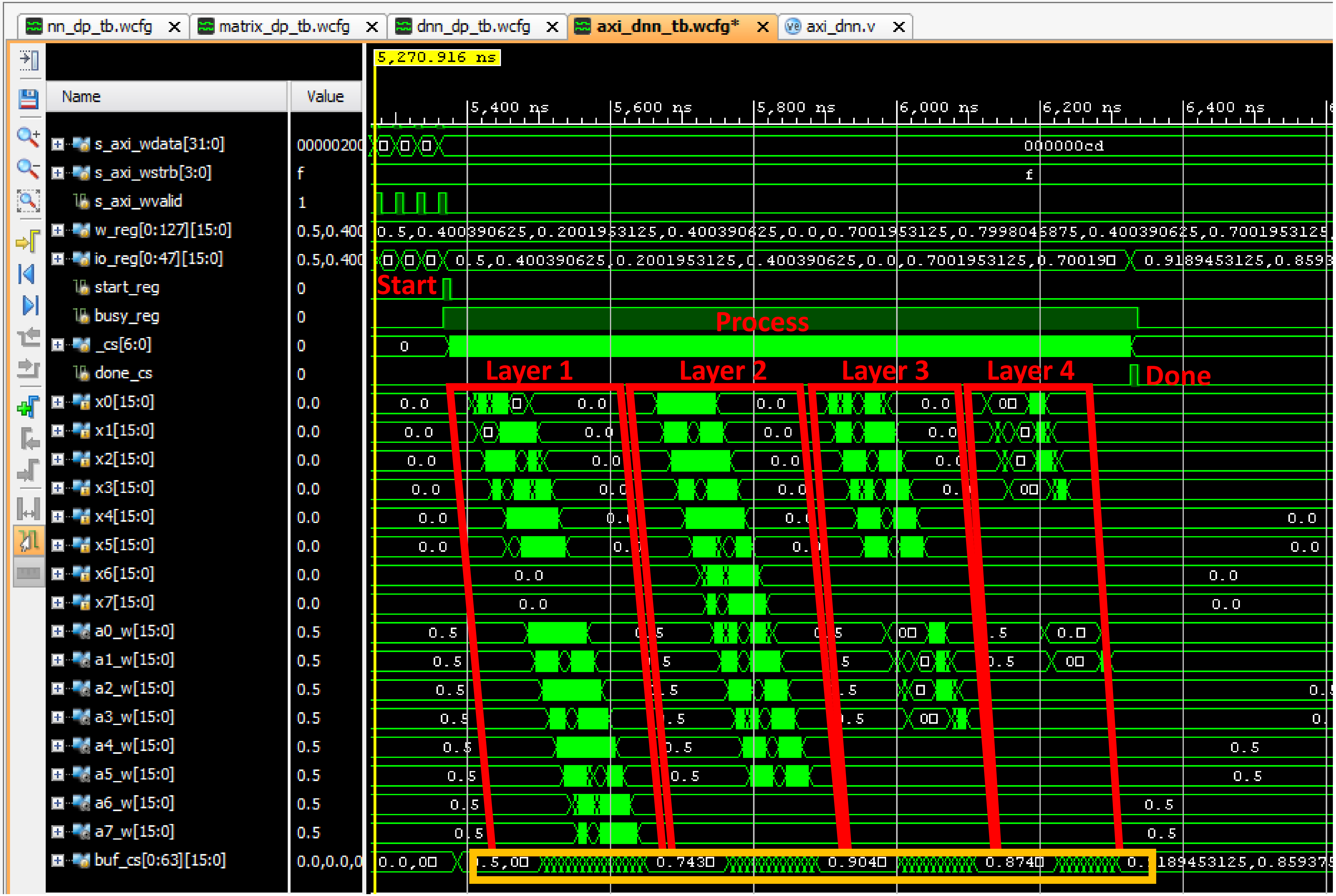
Total input=
6x8=48

Jumlah Register Buffer

- Jumlah register buffer berfungsi untuk menyimpan hasil perklaian matrix yang ukurannya menyesuaikan ukuran perkalian matrix.
- Karena ukuran perkalian matrix maximal 8x8, maka jumlah element matrix-nya adalah 8x8=64.
- Sehingga perlu 64 register untuk menyimpan output perkalian matrix.

State Machine untuk Controller

1. Menunggu sinyal START
2. Load input X dari IO reg ke buf reg
3. Load weight untuk layer n ke weight reg (register systolic)
4. Process perkalian input X dan weight layer n
5. Membaca output perkalian systolic yang sudah dilakukan fungsi sigmoid dan disimpan di buf reg
6. Ulangi ke step 3 sampai jumlah layer 4 terpenuhi
7. Menulis output dari buf reg ke IO reg
8. Generate sinyal DONE

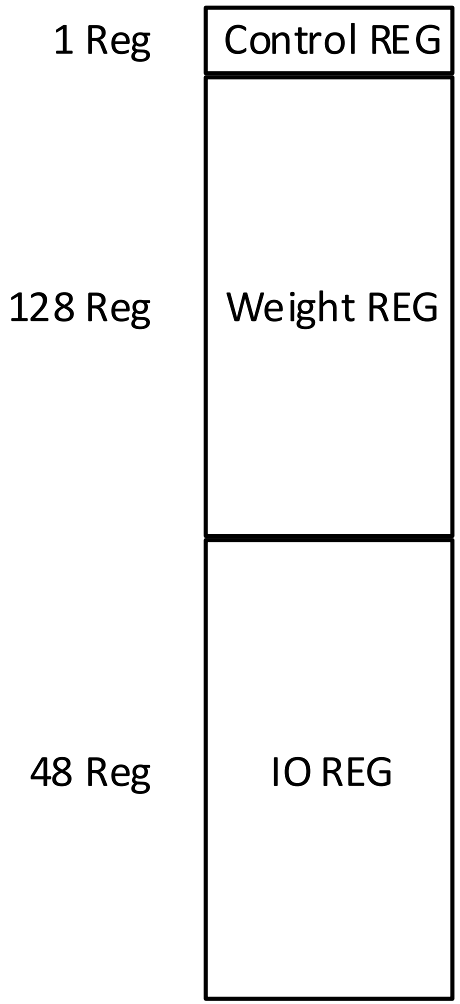


nn_dp_tb.wcfg | matrix_dp_tb.wcfg | dnn_dp_tb.wcfg | axi_dnn_tb.wcfg* | axi_dnn.v

Name	Value
s_axi_wdata[31:0]	00000200
s_axi_wstrb[3:0]	f
s_axi_wvalid	1
w_reg[0:127][15:0]	0.5,0.400
io_reg[0:47][15:0]	0.5,0.400
start_reg	0
busy_reg	0
_cs[6:0]	0
done_cs	0
x0[15:0]	0.0
x1[15:0]	0.0
x2[15:0]	0.0
x3[15:0]	0.0
x4[15:0]	0.0
x5[15:0]	0.0
x6[15:0]	0.0
x7[15:0]	0.0
a0_w[15:0]	0.5
a1_w[15:0]	0.5
a2_w[15:0]	0.5
a3_w[15:0]	0.5
a4_w[15:0]	0.5
a5_w[15:0]	0.5
a6_w[15:0]	0.5
a7_w[15:0]	0.5
buf_cs[0:63][15:0]	0.0,0.0,0.0

Address Register Map

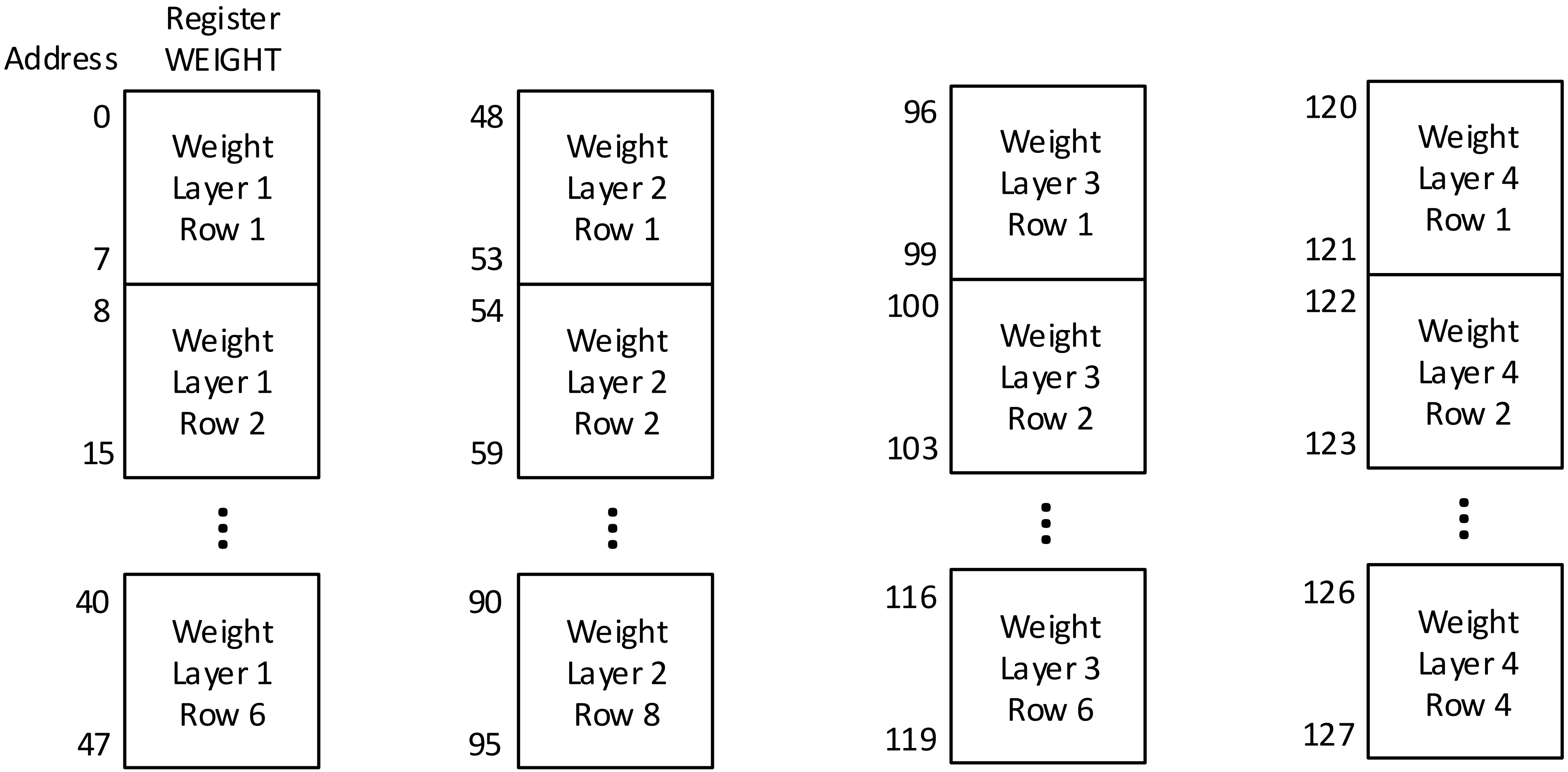
Address Map



Bit ke 0 : busy register
Bit ke 31-1: tidak digunakan

Start core secara otomatis berjalan ketika menulis data terakhir ke register IO alamat 48

Address Register Weight



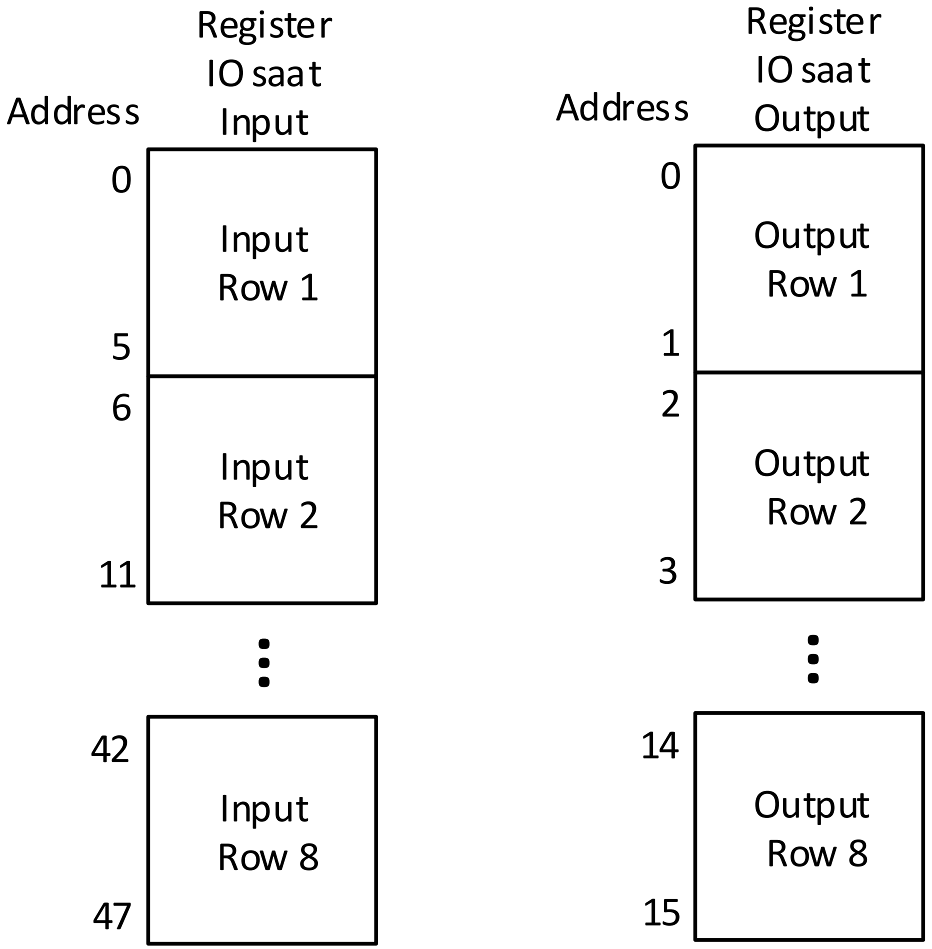
Address	Register WEIGHT
0 7	Weight Layer 1 Row 1
8 15	Weight Layer 1 Row 2
⋮	⋮
40 47	Weight Layer 1 Row 6

48 53	Weight Layer 2 Row 1
54 59	Weight Layer 2 Row 2
⋮	⋮
90 95	Weight Layer 2 Row 8

96 99	Weight Layer 3 Row 1
100 103	Weight Layer 3 Row 2
⋮	⋮
116 119	Weight Layer 3 Row 6

120 121	Weight Layer 4 Row 1
122 123	Weight Layer 4 Row 2
⋮	⋮
126 127	Weight Layer 4 Row 4

Address Register IO



Address

Register IO saat Input

0

Input Row 1

5

6

Input Row 2

11

⋮

42

Input Row 8

47

Address

Register IO saat Output

0

Output Row 1

1

2

Output Row 2

3

⋮

14

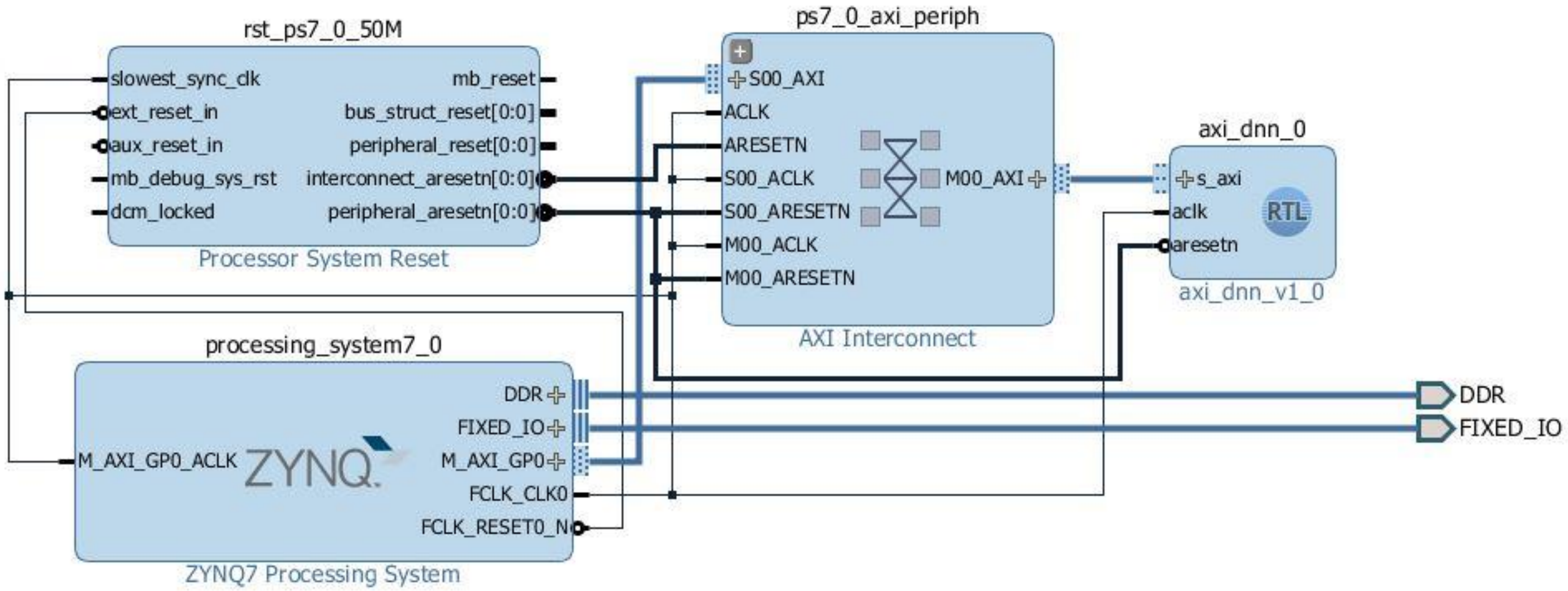
Output Row 8

15

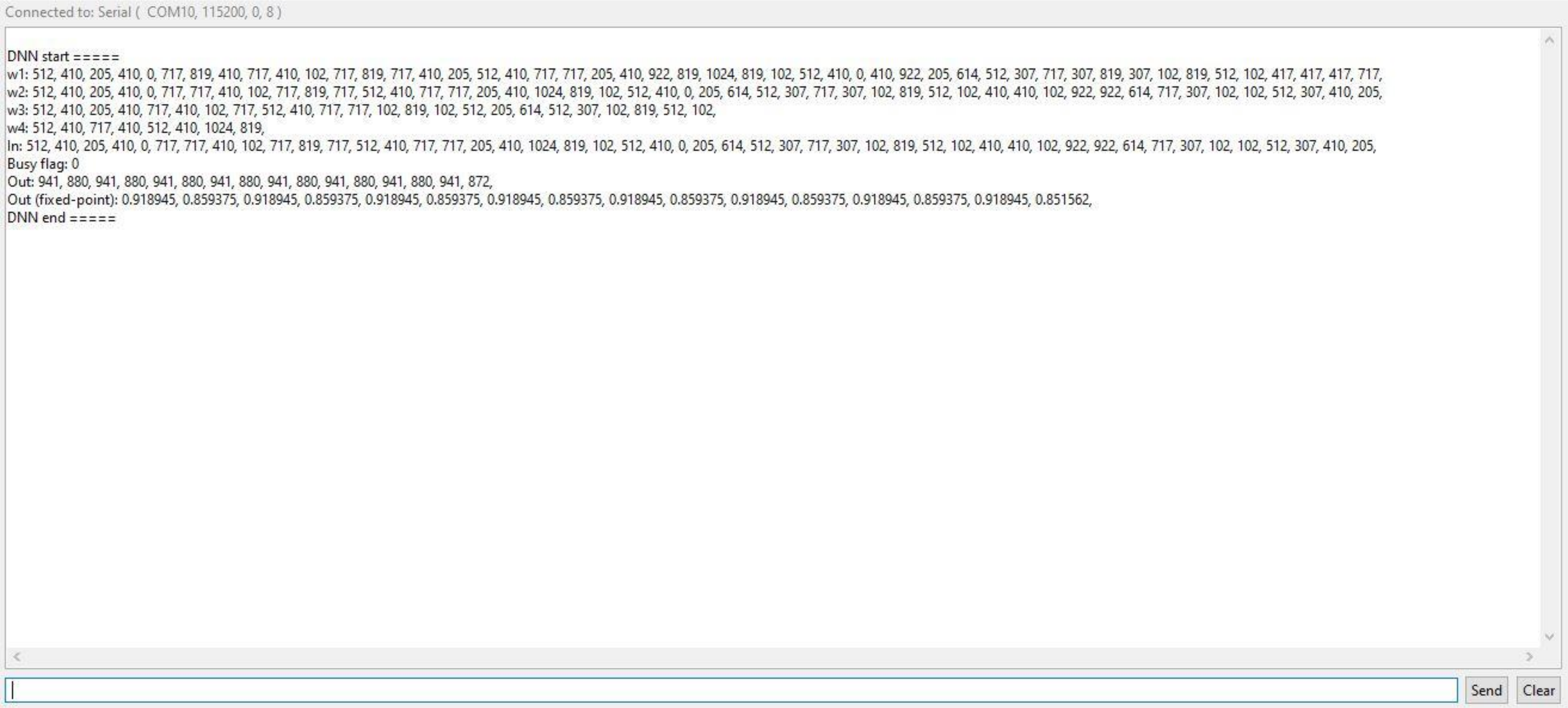
Implementasi di Board ZYBO

Block Design

- Tambahkan ZYNQ PS dan aktifkan UART 1
- Tambahkan module axi_dnn



Output SDK Terminal



Connected to: Serial (COM10, 115200, 0, 8)

```
DNN start =====
w1: 512, 410, 205, 410, 0, 717, 819, 410, 717, 410, 102, 717, 819, 717, 410, 205, 512, 410, 717, 717, 205, 410, 922, 819, 1024, 819, 102, 512, 410, 0, 410, 922, 205, 614, 512, 307, 717, 307, 819, 307, 102, 819, 512, 102, 417, 417, 417, 717,
w2: 512, 410, 205, 410, 0, 717, 717, 410, 102, 717, 819, 717, 512, 410, 717, 717, 205, 410, 1024, 819, 102, 512, 410, 0, 205, 614, 512, 307, 717, 307, 102, 819, 512, 102, 410, 410, 102, 922, 922, 614, 717, 307, 102, 102, 512, 307, 410, 205,
w3: 512, 410, 205, 410, 717, 410, 102, 717, 512, 410, 717, 717, 102, 819, 102, 512, 205, 614, 512, 307, 102, 819, 512, 102,
w4: 512, 410, 717, 410, 512, 410, 1024, 819,
In: 512, 410, 205, 410, 0, 717, 717, 410, 102, 717, 819, 717, 512, 410, 717, 717, 205, 410, 1024, 819, 102, 512, 410, 0, 205, 614, 512, 307, 717, 307, 102, 819, 512, 102, 410, 410, 102, 922, 922, 614, 717, 307, 102, 102, 512, 307, 410, 205,
Busy flag: 0
Out: 941, 880, 941, 880, 941, 880, 941, 880, 941, 880, 941, 880, 941, 880, 941, 872,
Out (fixed-point): 0.918945, 0.859375, 0.918945, 0.859375, 0.918945, 0.859375, 0.918945, 0.859375, 0.918945, 0.859375, 0.918945, 0.859375, 0.918945, 0.859375, 0.918945, 0.851562,
DNN end =====
```

Thank You