Integrated Circuits Design by FPGA

م.م. أحمد مؤيد عبدالحسين

جامعة الفرات الأوسط التقنية / الكلية التقنية الهندسية / نجف

Lecture 14

Neural Networks

Activation Functions

Objectives of this Lecture

- To define **Activation Functions**

- To study different types of **Activation Functions**.

Contents of this Lecture

- Introduction
- Activation Functions Types.

Introduction

- **Neural Networks** (NN) are highly parallel, highly interconnected systems. Such characteristics make their implementation very challenging, and also very costly, due to the large amount of hardware required.

- Artificial neural networks (ANNs) have been used successfully in solving pattern classification and recognition problems, function approximation and predictions. Their processing capabilities are based on their highly, parallel and interconnected architecture.

- A feedforward **NN** is shown in figure 12.12(a). In this example, the circuit has three layers, with three 3-input neurons in each layer. Internal details of each layer are depicted in figure 12.12(b).

Introduction

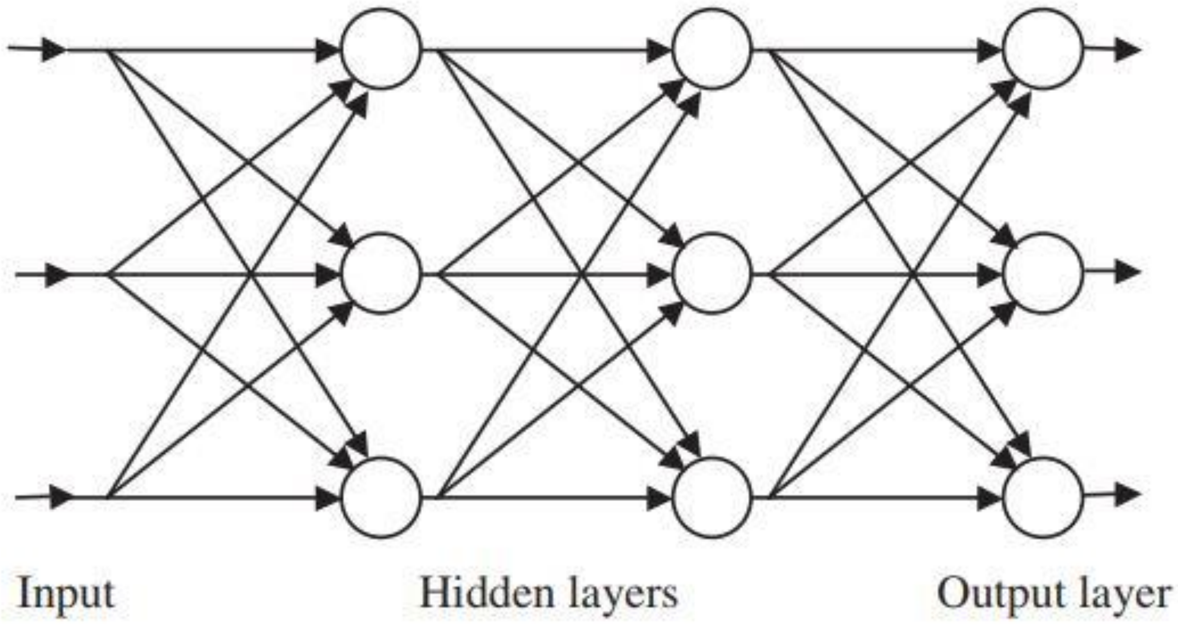


Figure : 12.12 a

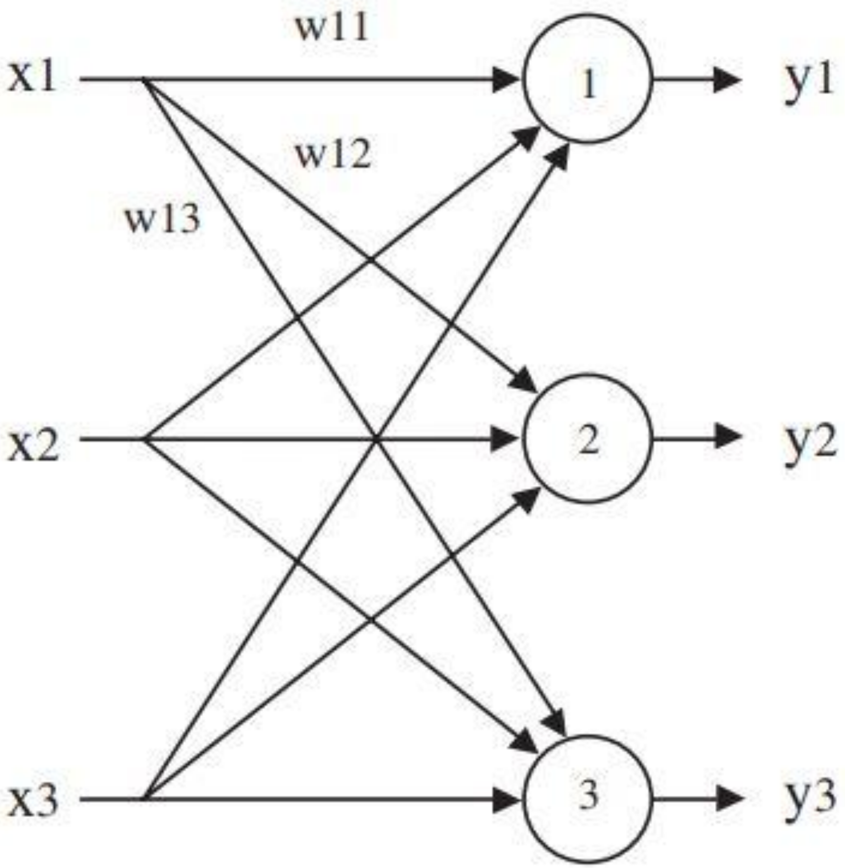
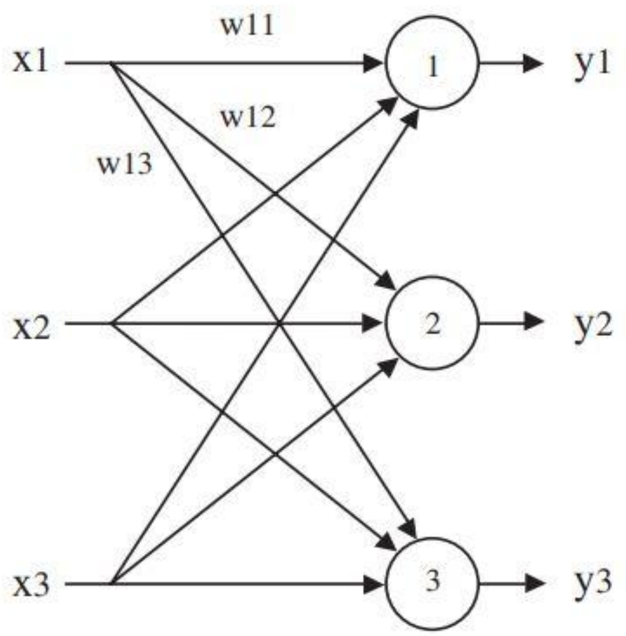


Figure : 12.12 b

Introduction

- $\mathbf{x_i}$ represents the **i**th input.
- $\mathbf{w_{ij}}$ is the weight between input **i** and neuron **j**,
- $\mathbf{y_j}$ is the **j**th output.
- Therefore, y1 = f(x1.w11 + x2.w21 + x3.w31),
- y2 = f(x1.w12 + x2.w22 + x3.w32),
- y3 = f(x1.w13 + x2.w23 + x3.w33),
- where f( ) is the **activation function** (linear threshold, sigmoid, etc.).



Introduction

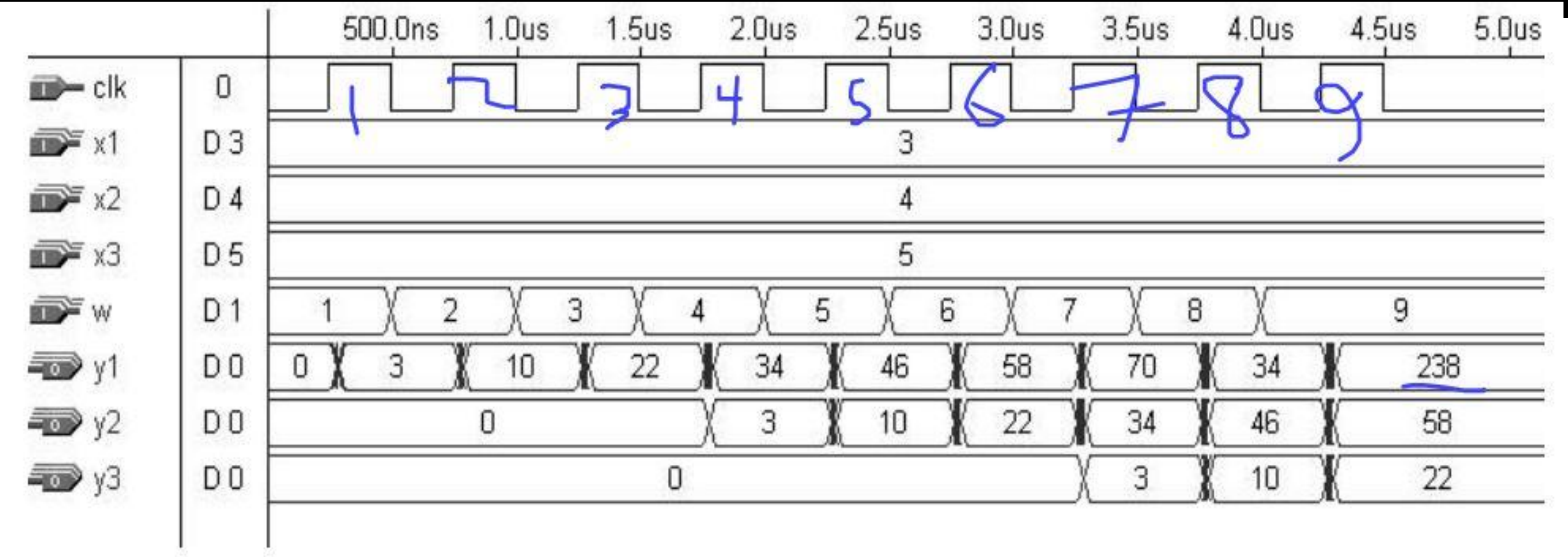
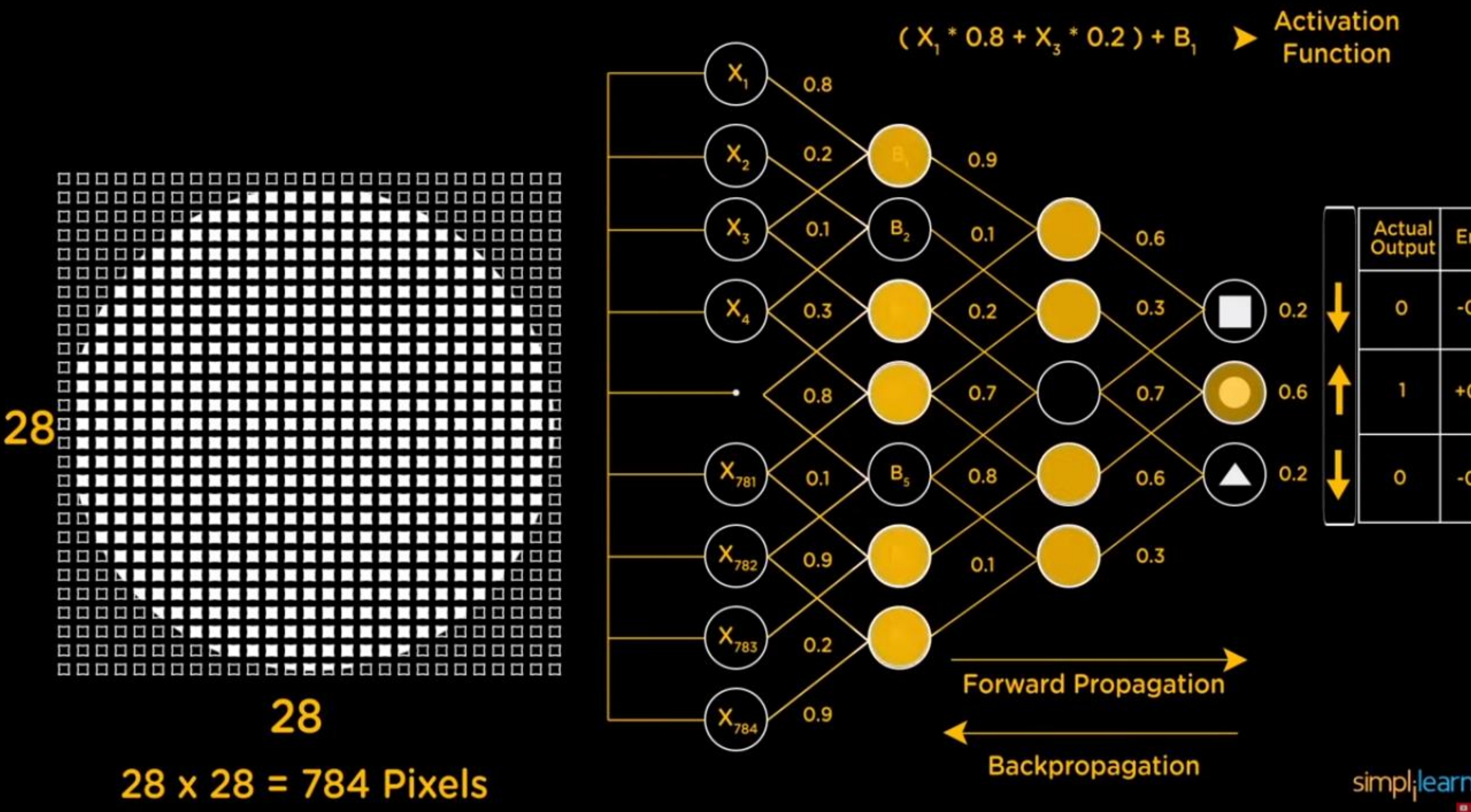


Figure 12.15
Simulation results of NN implemented in solution 1.

y1 = x1.w1 + x2.w2 + x3.w3 = (3)(−7) + (4)(−8) + (5)(7) = −18 (represented as 256 − 18 = 238); y2 = x1.w4 + x2.w5 + x3.w6 = (3)(6) + (4)(5) + (5)(4) = 58; and y3 = x1.w7 + x2.w8 + x3.w9 = (3)(3) + (4)(2) + (5)(1) = 22. These values (238, 58, and 22) can be seen at the right end of figure 12.15.



(X1 * 0.8 + X3 * 0.2) + B1
Activation Function
X1
X2
X3
X4
X781
X782
X783
X784
0.8
0.2
0.1
0.3
0.8
0.1
0.9
0.2
0.9
B2
B5
0.9
0.1
0.2
0.7
0.8
0.1
0.6
0.3
0.7
0.6
0.3
0.2
0.6
0.2
Actual Output
0
1
0
28
28
28 x 28 = 784 Pixels
Forward Propagation
Backpropagation

Activation Functions Types

1. Symmetrical Hard Limiter (hardlims).
2. Symmetric Saturating Linear (satlins).
3. Unit-Step function.
4. Ramp function.
5. Hyperbolic tangent sigmoid (tansig).

Note : A particular activation function of neuron is chosen to satisfy specification of the training algorithm that the neural network is attempted to run.

Activation Functions Types

1.) Symmetrical Hard Limit activation function (hardlims).

Used to classify input into two distinct categories.

$$a = \begin{cases} -1 & n \prec 0 \\ 1 & n \geq 0 \end{cases}$$

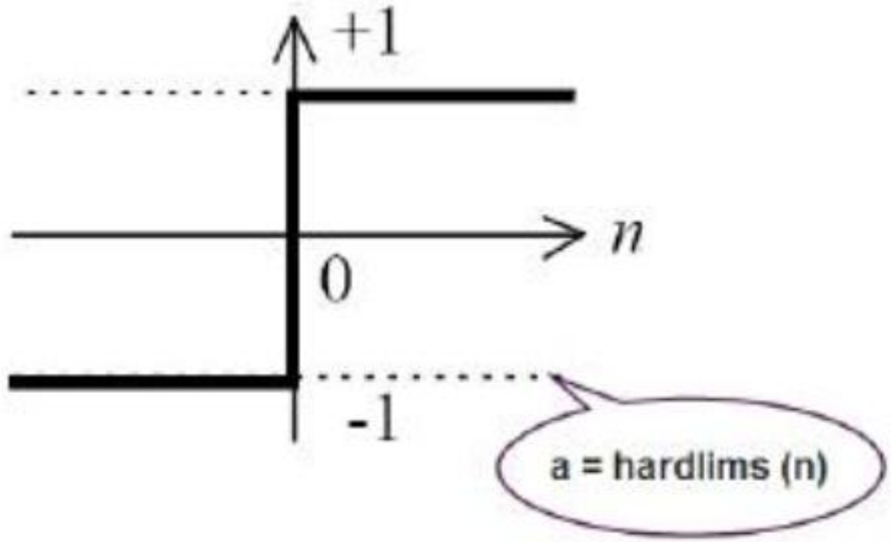


Fig.(4) Symmetrical hard limit activation function

Activation Functions Types

1.) Symmetrical Hard Limit activation function (hardlims).

hardlims function as VHDL package

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_SIGNED.ALL;
package hardlims_fun1 is
function hardlims  (signal n : signed) return signed;
end hardlims_fun1 ;
package body hardlims_fun1  is
function hardlims (signal n : signed) return signed is
variable a: signed(7 downto 0);
variable temp:integer range  -128 to 127;
begin
temp := conv_integer (n );
if ( temp >= 0 ) then  temp := 1 ;
else  temp := -1;
end if;
a <= conv_signed (temp,8);
return a;
end hardlims;
end hardlims_fun1;
```

8-bit signed neuron output

Activation Functions Types

2.) The saturating linear activation function (satlin)

$$a = \begin{cases} -1 & n \prec -1 \\ n & -1 \leq n \leq 1 \\ 1 & n \succ 1 \end{cases}$$

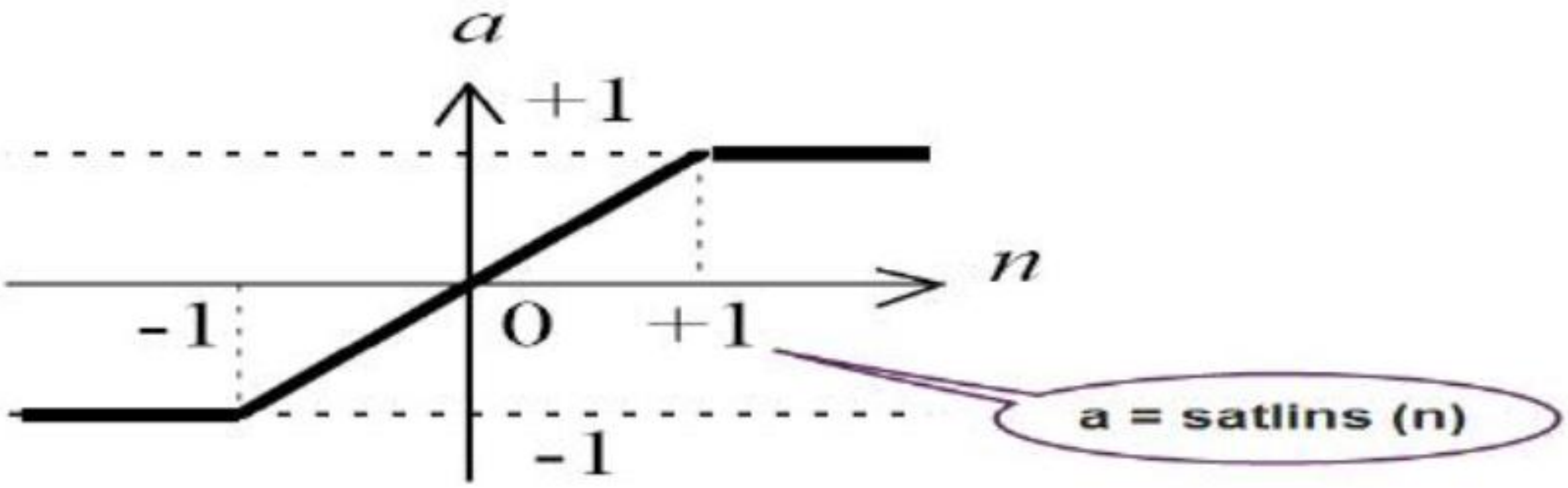


Fig. (6) Saturating linear activation function.

Activation Functions Types

2.) The saturating linear activation function (satlin)

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_SIGNED.ALL;
---------------------------------------------------------------
package satlins_fun2 is
function satlins  (signal nacc : signed) return signed;
end satlins_fun2 ;
---------------------------------------------------------------
package body satlins_fun2  is
function satlins  (signal nacc : signed) return signed is
variable aa: signed(7 downto 0);
variable sat1: signed(7 downto 0):= "01000000"; -- 64
variable sat2: signed(7 downto 0):= "11000000"; -- -64
begin
if ( nacc >= sat1 ) then aa := sat1  ;
elsif ( nacc <= sat2 ) then aa := sat2 ;
else aa := nacc ;
end if;
return aa;
end satlins;
end satlins_fun2;
```

VHDL code package for satlins function

saturation values

8-bit signed neuron output

Activation Functions Types

5.) Hyperbolic Tangent Sigmoid activation function (tansig)

The Hyperbolic tangent sigmoid (tansing) activation function is shown in Fig (8) . This function takes the input (**which may have any value between plus and minus infinity**) and the output value into the range - 1 to 1 , according to the expression:

$$a = \frac{e^{n} - e^{-n}}{e^{n} + e^{-n}}$$

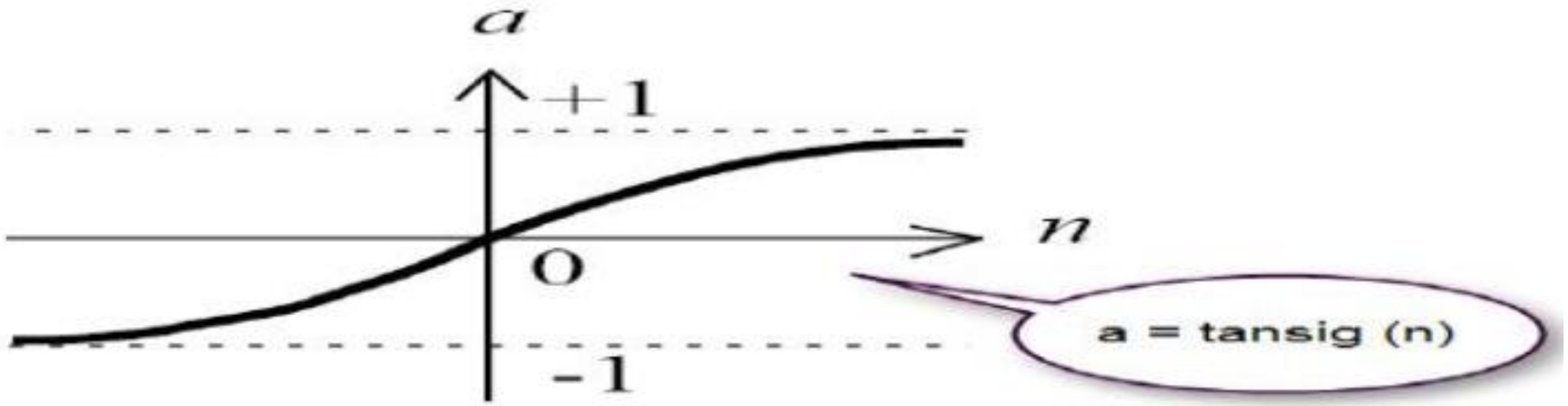


Fig.(8) Hyperbolic tangent sigmoid (tansig) activation function.

Activation Functions Types

5.) Hyperbolic Tangent Sigmoid activation function (tansig)

- **Note:** The **tansig** activation function is commonly used in multilayer neural networks that are trained by the back propagation algorithm.
- The tansig function is not easily implemented in digital hardware because it is consists of an infinite exponential series.
- Many researchers use a lookup table to implement the tansig function. The draw back of using lookup table is the great amount of hardware resources needed.
- A simple second order nonlinear function presented by Kwan, can be used as an approximation to a sigmoid function :

$$f(n) = \begin{cases} n(B - g.n) & for\ \ 0 \le n \le L \\ n(B + g.n) & for\ -L \le n \prec L \end{cases}$$

Assignments

The assignments will be attached to your class room

End of lecture 14
Any Questions ?