مقدمه

در فاز اول پروژه با طراحی توابع و مدارها با استفاده از هسته‌های مالکیت معنوی آشنا شدید. با توجه به پیچیدگی آن‌ها در این مرحله مدار توابع مورد نیاز در اختیار شما قرار داده شده است. لذا باید با استفاده از آن‌ها یک شبکه‌ی عصبی عمیق را توصیف نمایید. هدف از بخش آشنایی شما با طراحی در سطح انتقال ثبات و طراحی واحد کنترل است.
یک شبکه‌ی عصبی از دو بخش یادگیری و طبقه‌بندی تشکیل می‌شود. جهت کاهش پیچیدگی طراحی بخش یادگیری انجام شده و کافیست شما با پیاده‌سازی آن و انتساب ورودی به مدار داده‌ها را طبقه‌بندی نمایید.
در این شبکه‌ی عصبی ابتدا نام‌های انگلیسی و یونانی و نوع آن (انگلیسی یا یونانی بودن) به سیستم آموزش داده شده است و این سیستم با گرفتن نام فرد باید تشخیص دهد که نوع آن چیست. بدیهی است که نام ورودی می‌تواند در لیست نام‌های آموزش داده شده نباشد.

در هنگام تحویل حضوری باید فایل‌های ارسال‌شده در هر سه فاز به همراه داشته باشید تا در صورت درخواست ارائه دهید. نحوه‌ی تحویل هر فاز در ادامه آمده است.

فاز دوم	**تحویل غیرحضوری از طریق سامانه‌ی دروس تا ساعت ۲۳:۵۵ روز چهارشنبه مورخ ۹۸/۰۳/۸**
فاز سوم	**تحویل حضوری روز دوشنبه مورخ ۹۸/۰۴/۳**

چنانچه ابهامی در زمینه پروژه دارید، اشکالات خود را از طریق پست الکترونیکی زیر با موضوع DA.2019 رفع نمایید.

ali.mohammadpour@aut.ac.ir

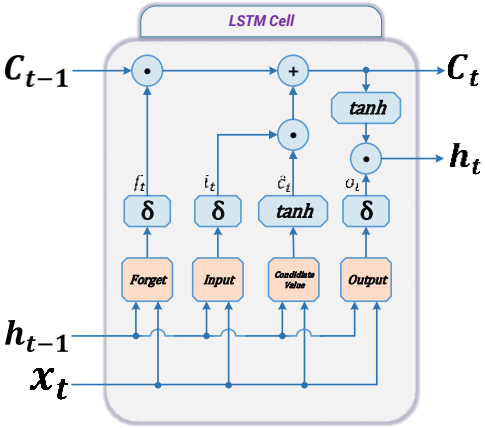
محمدپور

موفق و پیروز باشید!

فاز دوم

سلول LSTM

در این مرحله شما باید یک واحد پردازشی LSTM را طراحی نمایید. شکل ۱ نمای کلی و روابط این شبکه‌ی عصبی را نشان می‌دهد.



سلول LSTM

$$f_t = \delta(x_t \times W_f + h_{t-1} \times U_f + b_f)$$
$$i_t = \delta(x_t \times W_i + h_{t-1} \times U_i + b_i)$$
$$o_t = \delta(x_t \times W_o + h_{t-1} \times U_o + b_o)$$
$$\tilde{c}_t = \tanh(x_t \times W_c + h_{t-1} \times U_c + b_f)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$
$$h_t = \tanh(\tilde{c}_t) \circ o_t$$

روابط سلول LSTM

شکل ۱: سلول LSTM و روابط بین گره‌ها

در روابط شکل ۱

- منظور از علامت $\times$ ضرب دکارتی دو ماتریس است.
- منظور از علامت $\circ$ ضرب درایه به درایه دو ماتریس است.
- منظور از علامت $+$ جمع درایه به درایه دو ماتریس است.

جدول ۲-۱ نمادهای سلول

+	این مدار دو ماتریس را به عنوان ورودی گرفته و آن‌ها را درایه به درایه با هم جمع می‌کند.
•	این مدار دو ماتریس را به عنوان ورودی گرفته و آن‌ها را درایه به درایه در هم ضرب می‌کند.
δ	این مدار یک ماتریس را به عنوان ورودی گرفته و برای هر درایه حاصل $\boldsymbol{Sigmoid}$ را محاسبه می‌کند.
tanh	این مدار یک ماتریس را به عنوان ورودی گرفته و برای هر درایه حاصل $\boldsymbol{tanh}$ را محاسبه می‌کند.
Forget	این مدار مقدار $\boldsymbol{x_t \times W_f + h_{t-1} \times U_f + b_f}$ را محاسبه می‌کند.
Input	این مدار مقدار $\boldsymbol{x_t \times W_i + h_{t-1} \times U_i + b_i}$ را محاسبه می‌کند.
Condidiate Value	این مدار مقدار $\boldsymbol{x_t \times W_c + h_{t-1} \times U_c + b_c}$ را محاسبه می‌کند.
Output	این مدار مقدار $\boldsymbol{x_t \times W_o + h_{t-1} \times U_o + b_o}$ را محاسبه می‌کند.

توضیحات ماتریس‌ها و نوع داده

- در هر یک از بخش‌های سلول LSTM اندازه‌ی ماتریس‌ها و بردارها متفاوت است. اندازه‌ی هر یک از ماتریس‌ها در جدول ۲-۲ آمده است.

جدول ۲-۲ اندازه‌ی ماتریس‌ها و بردارها

نام	اندازه ماتریس	نام	اندازه ماتریس	نام	اندازه ماتریس
x_t	⟨۱ ۴⟩	h_{t-1}	⟨۱ ۸⟩	h_t	⟨۱ ۸⟩
W_f	⟨۴ ۸⟩	W_i	⟨۸ ۴⟩	W_o	⟨۴ ۸⟩
U_f	⟨۸ ۸⟩	U_i	⟨۸ ۸⟩	U_o	⟨۸ ۸⟩
b_f	⟨۸ ۱⟩	b_i	⟨۸ ۱⟩	b_o	⟨۱ ۸⟩
W_c	⟨۴ ۸⟩	c_{t-1}	⟨۱ ۸⟩	c_t	⟨۱ ۸⟩
U_c	⟨۸ ۸⟩	f_t	⟨۱ ۸⟩	o_t	⟨۱ ۸⟩
b_c	⟨۸ ۱⟩	i_t	⟨۱ ۸⟩	$\tilde{c}_t$	⟨۱ ۸⟩

- مفهوم و کاربرد هر یک از ماتریس‌ها (بردارها) در جدول ۲-۳ آمده است.

جدول ۲-۳ مفهوم ماتریس‌ها و بردارها

ماتریس یا بردار	توضیحات
x_t	این بردار، بردار ورودی مدار است که نشان‌دهنده‌ی یک حرف لاتین است.
h_{t-1} و c_{t-1}	این بردار، ورودی مدار است که از سلول قبلی می‌آید
c_t	این بردار، بردار خروجی است و به سلول بعدی می‌رود.
h_t	این بردار، بردار خروجی است و به سلول بعدی می‌رود.
سایر ماتریس‌ها	سایر ماتریس‌ها، ماتریس ضرایب و بایاس هستند که همیشه ثابت هستند و برای همه‌ی سلول‌ها مشترک است. این ماتریس‌ها می‌توانند در قالب یک حافظه‌ی ROM توصیف شوند.

- هر داده‌ی ماتریس (بردار) یک عدد ممیز شناور با دقت ساده است. لذا هر یک از درایه‌ها ۳۲ بیتی است. به عنوان مثال بردار x_t یک بردار ۸ تایی است که هر درایه‌ی آن ۳۲ بیت است.

```
TYPE matrix_1_4 IS ARRAY (0 To 3) Of STD_LOGIC_VECTOR(31 DOWNTO 0);
TYPE matrix_1_8 IS ARRAY (0 To 7) Of STD_LOGIC_VECTOR(31 DOWNTO 0);
TYPE matrix_4_8 IS ARRAY (0 To 3, 0 To 7) Of STD_LOGIC_VECTOR(31 DOWNTO 0);
TYPE matrix_4_8 IS ARRAY (0 To 7, 0 To 7) Of STD_LOGIC_VECTOR(31 DOWNTO 0);
```

ماژول‌های آماده

ماژول *tanh*

جهت پیاده‌سازی تابع $f(x) = \tanh(x)$ از tanh_module استفاده کنید. طراحی این ماژول خط‌لوله است و خروجی پس از چهار کلاک آماده می‌شود به هیچ عنوان ساختار کد را تغییر ندهید و برای پیاده‌سازی *tanh* مجاز به استفاده از ماژول دیگری نیستید.

```
ENTITY tanh_module is
    PORT (
            clk : IN STD_LOGIC;
            enable : IN STD_LOGIC;
            input : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
            output : OUT STD_LOGIC_VECTOR(31 DOWNTO 0));
END tanh_module;
```

ماژول *sigmoid*

جهت پیاده‌سازی تابع $g(x) = sigmoid(x)$ از tanh_module استفاده کنید. طراحی این ماژول خط‌لوله است و خروجی پس از شش کلاک آماده می‌شود. به هیچ عنوان ساختار کد را تغییر ندهید و برای پیاده‌سازی *sigmoid* مجاز به استفاده از ماژول دیگری نیستید.

```
ENTITY sigmoid_module IS
    PORT (
            clk : IN STD_LOGIC;
            enable : IN STD_LOGIC;
            input : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
            output : OUT STD_LOGIC_VECTOR(31 DOWNTO 0));
END sigmoid_module;
```

ماژول جمع ممیز شناور

جهت جمع دو عدد ممیز شناور از ماژول adder_module استفاده کنید. این ماژول دو عدد ممیز شناور ۳۲ بیتی را گرفته و یک عدد ممیز شناور ۳۲ بیتی برمی‌گرداند.

```
ENTITY add_module IS
    PORT (
            input : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
            output : OUT STD_LOGIC_VECTOR(31 DOWNTO 0));
END add_module;
```

ماژول ضرب ممیز شناور

جهت ضرب دو عدد ممیز شناور از ماژول multiplier_module استفاده کنید. این ماژول دو عدد ممیز شناور ۳۲ بیتی را گرفته و یک عدد ممیز شناور ۳۲ بیتی برمی‌گرداند.

```
ENTITY multiplier_module IS
    PORT (
            inputx : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
            inputy : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
            output : OUT STD_LOGIC_VECTOR(31 DOWNTO 0));
END multiplier_module;
```

مقادیر ماتریس‌های ضرایب و بایاس

محتوای ماتریس ضرایب همواره ثابت است. لذا بسته به نوع پیاده‌سازی شما می‌تواند به صورت یک حافظه با چندین پورت خروجی و یا با ساختار دیگر پیاده‌سازی شود. محتوای هر یک از ماتریس‌ها در قالب فایل ارائه شده است.

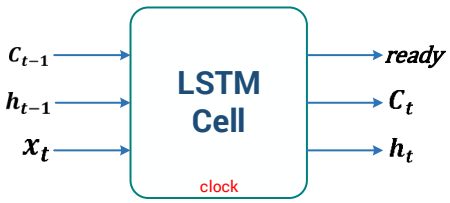
توابع آماده

جهت تبدیل نوع داده می‌توانید از Package طراحی شده استفاده کنید.

```
LIBRARY work;
USE work.neurals_utils.ALL;

    ---- Implemented Functions ----

    FUNCTION slv_to_single_float (
        input : IN STD_LOGIC_VECTOR(31 DOWNTO 0))
        return REAL;

    FUNCTION single_float_to_slv (
        input : IN REAL)
        RETURN STD_LOGIC_VECTOR;
```

شرح فاز دوم

با استفاده از ماژول‌ها، توابع و فایل ماتریس ضرایب یک سلول LSTM را در محیط XSIM (شبیه‌ساز ابزار Vivado) شبیه‌سازی کنید. ماژول سطح بالا (Top Module) ساختاری مطابق شکل ۲ دارد.



شکل ۲: سلول LSTM

```
ENTITY LSTM_Cell IS
    PORT (
        clk : IN STD_LOGIC;
        xt : IN matrix_1_4;
        ct_1 : IN matrix_1_8;
        ht_1 : IN matrix_1_8;
        ht : OUT matrix_1_8;
        ct : OUT matrix_1_8;
        ready : OUT STD_LOGIC );
END LSTM_Cell;
```

خروجی ready زمانی برابر یک می‌شود که نتیجه آماده شده باشد. بدیهی است که مدار فوق علاوه بر یک مسیرداده (Datapath) مانند شکل ۱، یک مدار کنترلی نیز دارد. بنابراین LSTM_Cell از دو ماژول Controller و Datapath تشکیل شده است. ممکن است مدار شما ورودی یا خروجی دیگری نیز داشته باشد.

نحوه ارزیابی فاز دوم

جهت ارزیابی یک فایل محیط آزمون (Test bench) در نظر بگیرید. سپس داده‌های هر یک از ورودی‌ها را مطابق فایل TestBenchDataSet0.txt و TestBenchDataSet1.txt بخوانید و نتایج را زمانی که ready برابر یک می‌شود در فایلی با نام TestBenchResult.txt بنویسید.

فایل‌کدهای نوشته‌شده (فقط فایل‌هایی با فرمت vhd) و فایل TestBenchResult.txt را در پوشه‌ای با نام src قرار دهید. یک فایل گزارش نیز بنویسید و در آن معماری خود را شرح دهید. در گزارش خود تعداد کلاک‌های مورد نیاز

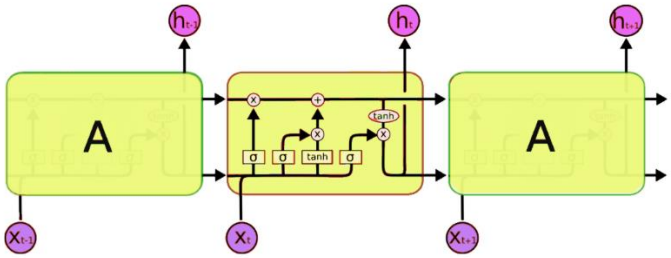
برای محاسبه و تعداد منابع استفاده شده را بنویسید. فایل‌های ذکر شده را تا موعد مقرر شده از طریق سامانه‌ی درس ارسال نمایید. کدها را در روز ارائه‌ی حضوری نیز بیاورید.

نکاتی در مورد نحوه‌ی پیاده‌سازی

با توجه به اینکه ماژول‌ها به صورت خط‌لوله (Pipeline) طراحی شده است، لذا می‌توانید مدار خود را بهبود بخشید. پیشنهاد می‌شود ابتدا مدار را مطابق روش فوق پیاده‌سازی کنید و حتی‌الامکان محاسبات ماتریسی (جمع و ضرب دکارتی و ضرب درایه‌ای) را به صورت موازی (Parallel) پیاده‌سازی کنید. سپس پس از گرفتن خروجی آن‌را به صورت خط‌لوله طراحی نمایید. طراحی خط‌لوله اجباری نیست ولی نمره‌ی اضافی خواهد داشت.

فاز سوم

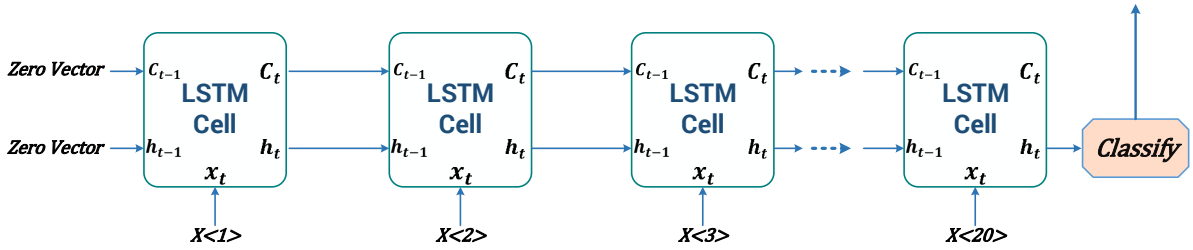
در این بخش با کنار هم قرار دادن سلول‌های LSTM شبکه‌ی عصبی کامل خواهد شد. شکل ۳ نمایی از یک شبکه‌ی LSTM را نشان می‌دهد.



شکل ۳: شبکه‌ی LSTM

پس از طراحی ماژول فاز دوم با کنار هم قرار دادن LSTM_CELL ها شبکه‌ی عصبی تکمیل می‌شود. در این مدار نیاز است تا ۲۰ سلول را در کنار هم دیگر قرار دهید.

- ورودی x_t برای هر یک از سلول‌ها متفاوت است.
- ماتریس ضرایب و بایاس همه‌ی سلول‌ها یکسان است.
- ورودی h_{t-1} خروجی h_t مرحله‌ی قبل است.
- ورودی c_{t-1} خروجی c_t مرحله‌ی قبل است.



شکل ۴: شبکه‌ی عصبی LSTM