การออกแบบระบบบนชิปและฮาร์ดแวร์เร่งประมวลผล LSTM แบบปรับแต่งได้

System on Chips Design with Configurable LSTM Hardware Accelerator

จัดทำโดย

นวพล ชิณวงศ์พรหม 63010507

ไพรสณฑ์ มูลเมือง 63010716

Nawaphon Chinwongprom 63010507

Prysol Moolmuang 63010716

รายงานนี้เป็นส่วนหนึ่งของวิชาการประยุกต์ทางอิเล็กทรอนิกส์ 2

ภาควิชาวิศวกรรมอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง พ.ศ.2565

การออกแบบระบบบนชิปและฮาร์ดแวร์เร่งประมวลผล LSTM แบบปรับแต่งได้

System on Chips Design with Configurable LSTM Hardware Accelerator

จัดทำโดย

นวพล ชิณวงศ์พรหม 63010507

ไพรสณฑ์ มูลเมือง 63010716

Nawaphon Chinwongprom 63010507

Prysol Moolmuang 63010716

รายงานนี้เป็นส่วนหนึ่งของวิชา project 2

ภาควิชาวิศวกรรมอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง พ.ศ.2565

รายงานวิชา การประยุกต์ทางอิเล็กทรอนิกส์ 1 ปีการศึกษา 2566

ภาควิชา วิศวกรรมอิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การออกแบบระบบบนชิปและฮาร์ดแวร์เร่งประมวลผล LSTM แบบปรับแต่งได้

System on Chips Design with Configurable LSTM Hardware Accelerator

ผู้จัดทำ นายนวพล ชิณวงศ์พรหม 63010507

นายไพรสณฑ์ มูลเมือง รหัสนักศึกษา 63010716

รายงานนี้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

(ผศ.ดร.สุเมฆ วิศยทักษิณ)

อาจารย์ที่ปรึกษา

หัวข้อโครงงาน การออกแบบระบบบนชิปและฮาร์ดแวร์เร่งประมวลผล LSTM แบบปรับแต่งได้

นักศึกษา นายนวพล ชิณวงศ์พรหม รหัสนักศึกษา 63010507

นายไพรสณฑ์ มูลเมือง รหัสนักศึกษา 63010716

ปริญญา วิศวกรรมศาสตร์บัณฑิต

ภาควิชา วิศวกรรมอิเล็กทรอนิกส์

ปีการศึกษา 2566

อาจารย์ที่ปรึกษาโครงงาน ผศ.ดร.สุเมฆ วิศยทักษิณ

บทคัดย่อ

โครงงานครั้งนี้เป็นส่วนหนึ่งของวิชา project 2 โดยจัดทำขึ้นเพื่อนำความรู้ในวิชาวิศวกรรมอิเล็กทรอนิกส์ที่ได้ศึกษานำมาสร้างและออกแบบระบบบนชิปสำหรับประมวลผลและรู้จำเสียงพูด โดยเป็นการออกแบบระบบบนชิปและฮาร์ดแวร์เร่งประมวลผล LSTM ที่สามารถปรับแต่งได้ และออกแบบหน่วยประมวลผลสัญญาณดิจิทัล เพื่อกรองข้อมูลผ่าน MFCC และศึกษาการวิเคราะห์ข้อมูลเพื่อจัดการให้ได้ระบบที่สามารถคำนวณเพื่อทำนายคำตอบของเสียงพูดได้ถูกต้อง

Project title	System on Chips Design with Configurable LSTM Hardware Accelerator
Students	Mr. Nawaphon Chinwongprom Student ID 63010507
	Mr. Prysol Moolmuang Student ID 63010716
Degree	Bachelor of Engineering
Program	Electronics Engineering
Academic Year	2023
Project Advisor	Asst. Prof. Sumek Wisayataksin

ABSTRACT

This project is a part of subject project2. It is prepare to bring knowledge in electronic engineering, use to create and design System-on-chip for Speech Recognition and Processing. and analyze its. Furthermore, this project also design the “Digital Signal Processing” using FPGA , So that the LSTM system can search data through MFCC. and study data analysis in order to get a system that makes the system can calculate to predicting the correct speech answer.

กิตติกรรมประกาศ

โครงงานนี้สามารถดำเนินการไปได้อย่างลุล่วงจาก ผศ.ดร สุเมฆ วิศยทักษิณ อาจารย์ที่ปรึกษาโครงงานที่ได้สละเวลาอันมีค่า เพื่อแนะนำ คำสอนต่างๆเกี่ยวกับความรู้ด้านวงจรดิจิทัล และการออกแบบปรับปรุงข้อมูล และแนะนำแนวทางแก้ไขปัญหาข้อบกพร่องต่างๆจนโครงงานนี้ดำเนินไปได้ด้วยดี ผู้จัดทำขอกราบขอบพระตุณเป็นอย่างสูงขอขอบคุณ อาจารย์ทุกท่าน และเพื่อนคณะวิศวกรรมศาสตร์ที่ช่วยให้คำแนะนำการทำโครงงานการใช้โปรแกรมในการตั้งค่าอุปกรณ์และการออกแบบการซื้อของทำโครงงานอำนวยสถานที่ในการทำโครงงานและแก้ไข ข้อผิดพลาดต่างๆเกี่ยวกับโครงงาน ทำให้โครงงานนี้ มีความคิบหน้าตามต้องการ และหวังว่าโครงงานเล่มนี้ จะเป็น ประโยชน์ต่อผู้อ่านไม่มากก็น้อย หากผิดพลาดประการใด ต้องขออภัยมา ณ ที่นี้ด้วย

นวพล ชิณวงศ์พรหม

ไพรสณฑ์ มูลเมือง

สารบัญ

สารบัญรูปภาพ

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ระบบบนชิป คือ การที่นำหน่วยประมวลผล หน่วยความจำ และส่วนของการควบคุมอื่นๆเข้ามารวมไว้อยู่ภายในชิปตัวเดียว ซึ่งก่อให้เกิดเป็นวงจรชิปที่ถูกออกแบบมาสำหรับการใช้งานใดการใช้งานหนึ่งโดยเฉพาะ (Application Specific Integrated Circuit) หรืออาจเป็นหน่วยประมวลผลทั่วไป (General Purposed Processor) สัญญาณเสียงที่ใช้ในการสื่อสาร มีลักษณะของคำพูดแต่ละคำที่เป็นเอกลักษณ์ ซึ่งในคำพูดแต่ละคำจะมีลักษณะความเข้มของเสียงต่างกันในแต่ละช่วง (Amplitude) เราสามารถใช้สัญญาณเสียงนี้เป็นข้อมูลมาใช้ในการวิเคราะห์และจำแนกออกมาเป็นคำสั่งที่ชัดเจนในโครงงานนี้ จึงสร้างระบบบนชิปและฮาร์ดแวร์เร่งประมวลผล LSTM แบบปรับแต่งได้มาใช้เพื่อประมวลผลการรู้จำเสียงพูด และ สามารถเข้าใจในคำสั่งเสียงผ่านไมโครโฟนเพื่อทำตามคำสั่งที่พูดโดยระบบบนชิปสามารถยืดหยุ่นที่จะปรับค่าความละเอียดและความเสถียรของระบบเพื่อให้เหมาะสมกับการใช้งานได้อย่างมีประสิทธิภาพโดยจะแบ่งออกได้เป็น 2 ส่วนในโครงงานชิ้นนี้เป็นส่วนแรก

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อศึกษาการสร้างหน่วยประมวลผลเฉพาะทาง LSTM

1.2.2 เพื่อสร้างระบบบนชิปเพื่อประมวลผลเสียง

1.3 ขอบเขตของการศึกษา

1.3.1 ออกแบบระบบบนชิปด้วย Verilog Hardware Description Language

1.3.2 ออกแบบระบบบนชิประบบบนชิปและฮาร์ดแวร์เร่งประมวลผล LSTM แบบปรับแต่งได้โดยสามารถทำงานได้อย่างถูกต้องบน simulation

1.4 ระยะเวลาในการทำโครงงาน

ตั้งแต่วันที่ 10 สิงหาคม 2566 ถึงวันที่ 10 มีนาคม 2566

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

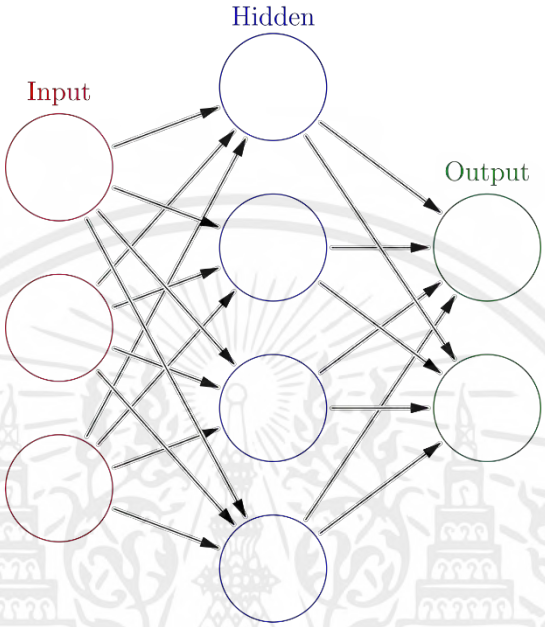
2.1 Artificial neural networks

โครงข่ายประสาทเทียม (Artificial neural networks: ANN) หรือเรียกได้ว่า ข่ายงานประสาทเทียม (Connectionist systems) คือระบบคอมพิวเตอร์จากโมเดลทางคณิตศาสตร์ เพื่อจำลองการทำงานโครงข่ายประสาทชีวภาพที่อยู่ในสมองของสัตว์ โครงข่ายประสาทเทียมสามารถเรียนรู้ที่จะทำงานที่มอบหมายได้ จากการเรียนรู้ผ่านตัวอย่าง โดยไม่ถูกโปรแกรมด้วยกฎเกณฑ์ตายตัวแบบระบบอัตโนมัติ ยกตัวอย่างเช่น ในการประมวลผลภาพ คอมพิวเตอร์ที่ทำงานด้วยระบบโครงข่ายประสาทเทียมจะเรียนรู้การจำแนกรูปภาพแมวได้จากการให้ตัวอย่างรูปภาพที่กำกับโดยผู้เขียนโปรแกรมว่า “เป็นแมว” หรือ “ไม่เป็นแมว” จากนั้นนำผลลัพธ์ที่ได้ไปใช้ระบุภาพแมวในตัวอย่างรูปภาพอื่น ๆ โปรแกรมโครงข่ายประสาทเทียมสามารถแยกแยะรูปภาพแมวได้โดยปราศจากการความรู้ก่อนหน้า ว่า ”แมว” คืออะไร (อาทิ แมวมีขน มีหูแหลม มีเขี้ยว มีหาง) แทนที่จะใช้ความรู้ดังกล่าว โครงข่ายประสาทเทียมทำการระบุตัวแมวโดยอัตโนมัติด้วยการระบุลักษณะเฉพาะ จากชุดตัวอย่างที่เคยได้ประมวลผล

แนวคิดเริ่มต้นของเทคนิคนี้ได้มาจากการศึกษาโครงข่ายไฟฟ้าชีวภาพ (bioelectric network) ในสมอง ซึ่งประกอบด้วย เซลล์ประสาท (neurons) และ จุดประสานประสาท (synapses) ตามโมเดลนี้ข่ายงานประสาทเกิดจากการเชื่อมต่อระหว่างเซลล์ประสาท จนเป็นเครือข่ายที่ทำงานร่วมกัน

การประมวลผลต่าง ๆ ของโครงข่ายประสาทเทียมเกิดขึ้นในหน่วยประมวลผลย่อยเรียกว่า โหนด (node) ซึ่งโหนดเป็นการจำลองลักษณะการทำงานมาจากเซลล์การส่งสัญญาณ ระหว่างโหนดที่เชื่อมต่อกันจำลองมาจากการเชื่อมต่อของใยประสาท และแกนประสาทในระบบประสาทของสมองมนุษย์ภายในโหนด จุดเชื่อมต่อแต่ละจุด มีความคล้ายคลึงกับจุดประสานประสาท (Synapses) ในสมอง มีความสามารถในการส่งสัญญาณไปยังเซลล์ประสาทเซลล์อื่น ๆ ที่เชื่อมต่อกับมันได้

การสร้างระบบโครงข่ายประสาทเทียมรูปแบบพื้นฐานเพื่อใช้ในการอธิบาย architecture ของระบบโครงข่ายประสาทเทียม จะพูดถึง Feed-Forward Neural Networks โดยมีส่วนประกอบอยู่ 3 layers ได้แก่ Input Layer, Hidden Layer และ Output Layer



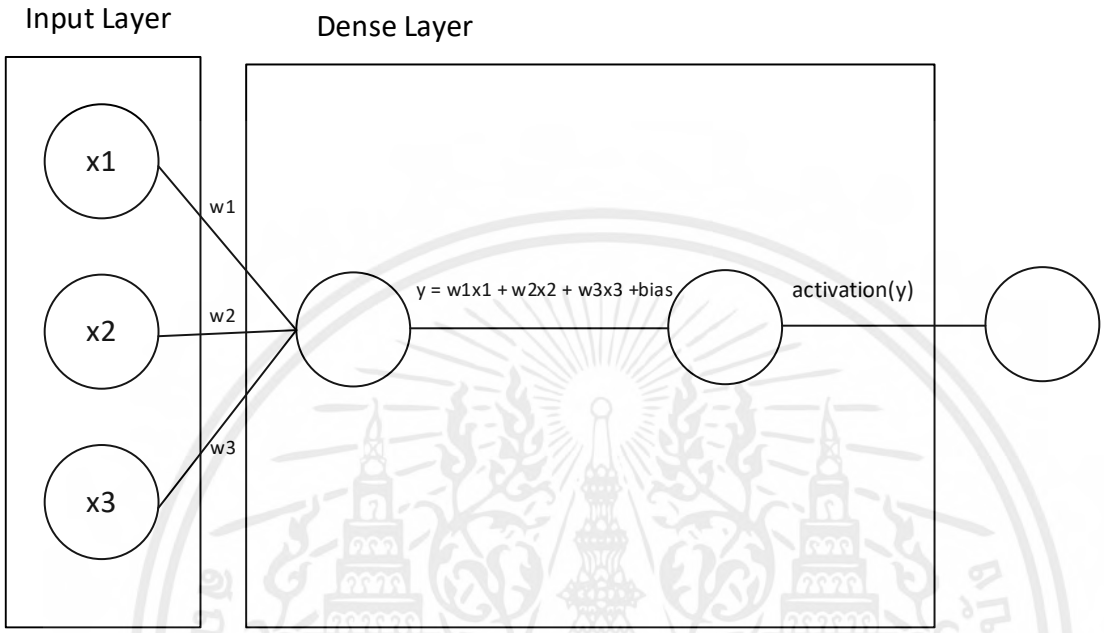
รูปที่ 2.1.1 Feed-Forward Neural Networks Architecture

ส่วนประกอบของ layer

- Input Layer เป็นชั้นที่ข้อมูลถูกส่งเข้ามาในระบบโครงข่ายประสาทเทียมเพื่อนำมาประมาลผล
- Hidden Layer เป็นชั้นที่จะนำข้อมูลที่ได้จาก Input Layer เพื่อนำมาใช้ในการคำนวณตามลักษณะของ Layer ภายใน เช่น Dense และ LSTM เป็นต้น
- Output Layer เป็นทำหน้าที่ในการทำนายผลจากการคำนวณข้อมูลจาก Hidden Layer ชั้นสุดท้าย

2.2 Dense Layer

ชั้น Dense เป็นชั้นที่ไว้ใช้ในการเชื่อมต่อกับข้อมูลของชั้นก่อนหน้าเพื่อนำข้อมูลมาคำนวนในลักษณะของสมการ Linear Regression และใช้งาน activation function ก่อนส่งข้อมูลที่ได้ออกไป



รูปที่ 2.2.1 Dense Layer

จากรูปที่ 2.2.1 จะเห็นว่า 1 neurons ของชั้น Dense เป็นการรวมของข้อมูลจากชั้น Input ทั้งหมดมาใส่ในสมการ Linear Regression ดังต่อไปนี้

$$y = w_1x_1 + w_2x_2 + w_3x_3 + \text{bias} \quad \text{.....(1)}$$

โดยที่ w_n คือค่า weight ของ input ตัวที่ n โดยค่านี้ไว้ใช้ในการปรับลดค่าของ input เพื่อให้ตัว neural network ของเรามีความแม่นยำมากขึ้น

x_n คือข้อมูลจากชั้น Input ที่ neurons ที่ n

bias คือค่าคงที่เพื่อไว้เพื่อให้ตัว neural network ของเรามีความแม่นยำขึ้น

ทั้งนี้เราจะเห็นได้จากสมการว่าการปรับ w_n และ bias ซึ่งเป็นค่าคงที่ของ neurons แต่ละตัวภายใน ชั้น Dense ส่งผลต่อความแม่นยำในการวิเคราะห์ข้อมูล

สุดท้ายก่อนที่จะส่ง y ที่ได้จากการคำนวณไปยังชั้นถัดไปจะต่องนำมาผ่านตัวของ activation function ก่อนเพื่อปรับให้ข้อมูลที่จะส่งไปอยู่ในอัตราส่วนที่เหมาะสม โดย activation function สามารถเลือกใช้ได้เช่น ReLU ,sigmoid และ tanh เป็นต้น

2.3 Activation function

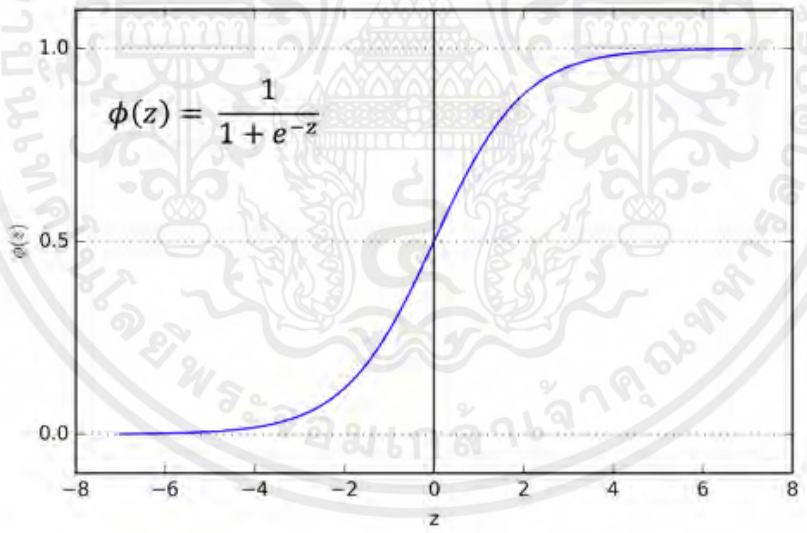
Activation Function คือ ฟังก์ชันที่รับผลรวมการประมวลผลทั้งหมด จากทุก Input (ทุก Dendrite ภายใน 1 นิวรอน แล้วพิจารณาว่าจะส่งต่อเป็น Output เท่าไรมีตัวอย่างดังนี้

1. Sigmoid function

เป็นฟังก์ชันที่เป็น Curve รูปตัว S เห็นแล้วเข้าใจได้ง่าย และเนื่องจาก Output ของ Sigmoid Function มีค่าระหว่าง 0 – 1 จึงเหมาะที่ sigmoid function จะถูกใช้ในงานที่ต้องการ Output เป็นความน่าจะเป็น (Probability) หรือใช้เป็น Output ว่า 1=Yes, 0=No เขียนเป็นสมการได้ดังนี้

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^{x}}{e^{x}+1} \quad \text{.....(2)}$$

กราฟจากสมการ Sigmoid function



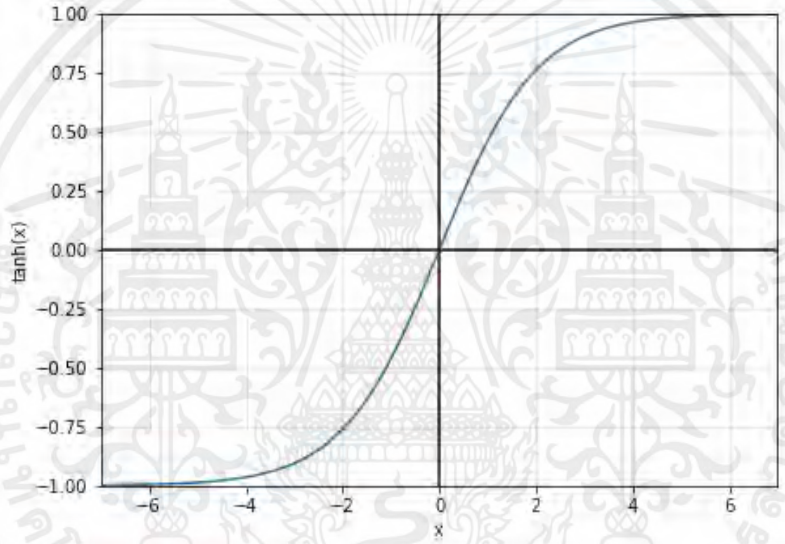
รูปที่ 2.3.1 Sigmoid function

2. Tanh function

เป็นฟังก์ชันที่เป็น Curve รูปตัว S เห็นแล้วเข้าใจได้ง่าย และเนื่องจาก Output ของ tanh Function มีค่าระหว่าง -1 – 1 ลักษณะคล้ายกับ sigmoid function Output มีความ Balance มี Mean เท่ากับ 0 จะทำให้สามารถ Optimize ข้อมูลได้ง่ายขึ้นเขียนเป็นสมการได้ดังนี้

$$f(x) = \tanh(x) = \frac{(e^{x}-e^{-x})}{(e^{x}+e^{-x})} \quad \text{.....(3)}$$

กราฟจากสมการ Tanh function



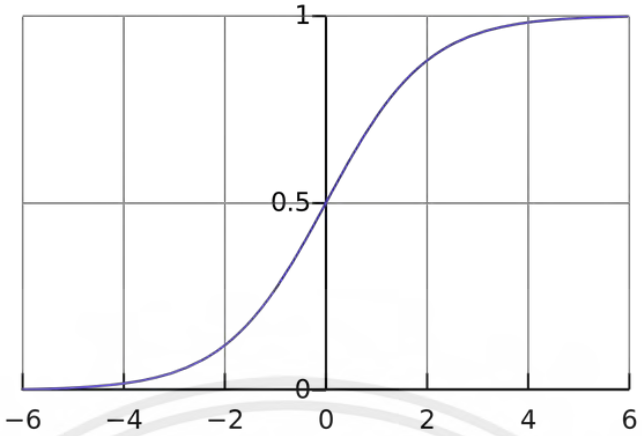
รูปที่ 2.3.2 Tanh function

3. Softmax function

Softmax Function หรือ SoftArgMax Function หรือ Normalized Exponential Function คือ ฟังก์ชันที่รับ Input เป็น Vector ของ Logit จำนวนจริง แล้ว Normalize ออกมาเป็นความน่าจะเป็น Probability ที่ผลรวมเท่ากับ 1

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad for\ j=1,\ldots,K \quad \text{.....(4)}$$

กราฟจากสมการ Softmax function



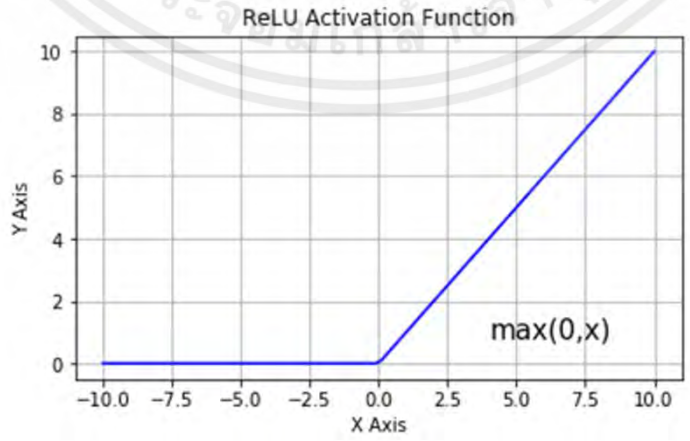
รูปที่ 2.3.3 Softmax function

4. Relu function

ReLU ย่อมาจาก Rectified Linear Unit คือ ฟังก์ชันที่ถูกปรับแก้ Rectified มีลักษณะเป็นเส้นตรงเฉียงขึ้นไปด้านขวา

$$f(x) = \max(0, x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases} \quad \text{.....(5)}$$

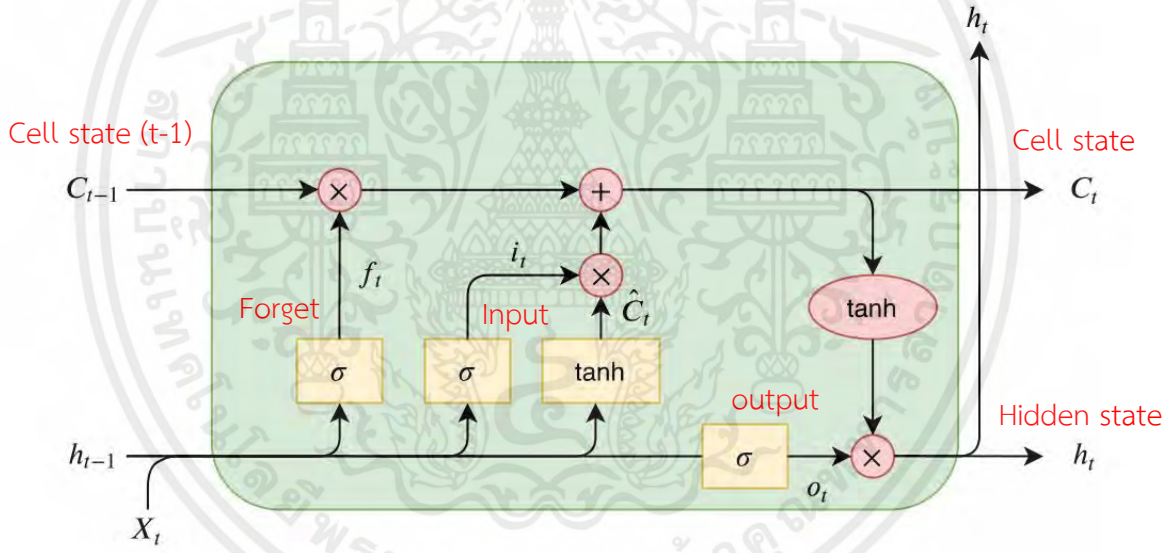
กราฟจากสมการ Relu function



รูปที่ 2.3.4 Relu function

2.4 Long Short-Term Memory Layer (LSTM Layer)

Hochreiter และ Schmidhuber ได้นำเสนอถึง Recurrent Neural Network (RNN) ชนิดใหม่ซึ่งมีชื่อว่า Long Short-Term Memory (LSTM) ซึ่งได้นำมาช่วยในการแก้ไขปัญหา Vanishing Gradient ที่ถูกพบเมื่อใช้วิธี RNN กับข้อมูลที่มีความยาว เช่น เสียงพูดหรือ และวีดิทัศน์ เป็นต้น โดย LSTM ประกอบด้วย input gate, output gate และ forget gate ซึ่งจะเป็นสิ่งที่ควบคุมการไหล ของข้อมูล โดยที่เมื่อ LSTM ได้รับข้อมูลมาจากชั้น input เป็นครั้งแรก LSTM จะเข้าสู่ input gate และเข้าสู่ output gate เพื่อตัดสินใจว่าจะเก็บค่าที่ได้ไว้แล้วจะวนซ้ำใน LSTM หรือแสดงผลข้อมูล หากเลือกที่จะวนซ้ำจะนำข้อมูลกลับมาเพื่อเข้าสู่ forget gate ซึ่งจะตัดสินใจว่าจะลบค่าที่เก็บไว้ทิ้ง หรือยังคงเก็บค่าไว้ หากเก็บไว้จะไปรอการอัพเดทจาก input gate ซึ่งจะตัดสินใจว่าจะอัพเดทค่านั้น หรือไม่ และจะอัพเดทด้วยค่าอะไร แล้วส่งค่านั้นไป output gate เพื่อตัดสินใจว่าจะนำข้อมูลนั้น ออกไปแสดงหรือว่านำข้อมูลกลับไปวนซ้ำอีกรอบ ดังนั้น LSTM จึงสามารถเรียนรู้จากข้อมูลที่เป็น ลำดับและเก็บหรือลบข้อมูลทิ้งถ้าข้อมูลนั้นไม่จำเป็น



รูปที่ 2.4.1 ภาพประกอบการทำงานของ LSTM

2.4.1 Cell state (C_t) เป็นตัวเก็บ state ของ memory cell ใน LSTM โดยจะทำการอัปเดตข้อมูลผ่าน input gate และ forget gate

2.4.2 Gate เป็นตัวที่ควบคุมการไหลของข้อมูลค่า analog ที่คอยควบคุมว่าควรจะอ่านข้อมูล read, write หรือ forget ลืมข้อมูลโดยทำหน้าที่คัดกรองว่าเวลาไหนควรเปิดรับข้อมูลหรือไม่สนใจข้อมูล แล้วปล่อยให้ข้อมูลไหลผ่านไป (forget) โดยใน (LSTM) จะประกอบไปด้วย gate ดังนี้

1. Forget

การ forget คือการล้างข้อมูลใน cell state เดิมที่เก็บค่าไว้ออกไป เตรียมล้างจัดการพื้นที่ว่างเพื่อรับข้อมูลชุดใหม่ โดยตัดสินใจว่าจะล้างข้อมูลหรือเก็บไว้คือตัว forget gate ที่จะควบคุมการไหลของข้อมูลถ้า forget gate เป็น 0 จะทำการล้างข้อมูล ส่วนถ้ามีค่าเท่ากับ 1 จะทำการเก็บข้อมูลของ cell state นี้

การสร้าง forget gate นี้ เราจะดู input data ที่เข้ามา ประกอบกับ hidden state ก่อนหน้า (ตามสูตรของ RNN) ประกอบการตัดสินใจ โดยจะใช้ sigmoid function เป็นตัวตัดสิน ดังสมการ

$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f + b^f\right) \quad \text{.....(6)}$$

σ คือ sigmoid function
x_t คือค่าอินพุทขาเข้า
h_{t-1} คือค่า hidden state ก่อนหน้า
U^f คือค่า weight ของ forget gate
W^f คือค่า weight ของ hidden state
b^f คือค่า bias ของ forget gate

2. Write

เมื่อมี input data ใหม่เข้ามาโดยในส่วนนี้เราจะพิจารณาว่าจะอัปเดตค่าใน cell state ด้วยข้อมูลชุดใหม่หรือไม่ และจะอัปเดตด้วยข้อมูลอะไร โดยการอัปเดตค่านั้นจะถูกควบคุมโดยสิ่งที่เรียกว่า input gate โดยใช้ sigmoid function เป็นตัวการตัดสินใจว่าจะอณุญาตให้มีการอัปเดตข้อมูลหรือไม่ โดยในการคำนวณจะใช้ค่า input data ที่รับมากับ hidden state ของค่าก่อนหน้านั้นเขียนเป็นสมการได้ดังนี้

$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i + b^i\right) \quad \text{.....(7)}$$

σ คือ sigmoid function
x_t คือค่าอินพุทขาเข้า
h_{t-1} คือค่า hidden state ก่อนหน้า
U^i คือค่า weight ของ input gate
W^i คือค่า weight ของ hidden state
b^i คือค่า bias ของ input gate

หลังจากที่เลือว่าจะอัปเดตข้อมูลผ่าน sigmoid function แล้วเราจะใช้สิ่งที่เรียกว่า modulation gate เป็นตัวควบคุมคล้ายๆกับ sigmoid function แต่จะใช้เป็นสมการของ tanh function แทนดังนี้

$$\tilde{C}_t = tanh(x_t U^g + h_{t-1} W^g + b^c) \quad \text{.....(8)}$$

$tanh$ คือ tanh function

x_t คือค่าอินพุทขาเข้า

h_{t-1} คือค่า hidden state ก่อนหน้า

U^g คือค่า weight ของ modulation gate

W^g คือค่า weight ของ hidden state

b^c คือค่า bias ของ modulation gate

3. อัพเดท cell state

การอัปเดตค่าภายใน cell state ทำได้โดยการรวมสมการของ forget gate, input gate, input modulation gate จะได้ออกมาเป็นสมการของ cell state ดังนี้

$$C_t = \sigma(f_t * C_{t-1} + i_t * \tilde{C}_t) \quad \text{.....(9)}$$

σ คือ sigmoid function

f_t คือค่าจาก forget gate

C_{t-1} คือค่า cell state ก่อนหน้า

i_t คือค่าจาก input gate

$\tilde{C}_t$ คือค่าจาก modulation gate

ในส่วนของสมการชุดแรกในการอัปเดตค่า cell state

โดย f_t forget gate มีค่าเป็น 0 จะไม่สนใจค่า C_{t-1} จากข้อมูลชุดที่แล้ว

f_t forget gate มีค่าเป็น 1 จะสนใจค่า C_{t-1} จากข้อมูลชุดที่แล้วมาใช้อัปเดต C_t ปัจจุบัน

ในส่วนของสมการชุดที่สองในการอัปเดตค่า cell state

โดย i_t input gate มีค่าเป็น 0 จะไม่สนใจค่า g_t ข้อมูลที่เตรียมไว้รออัปเดต

i_t input gate มีค่าเป็น 1 จะสนใจค่า g_t ข้อมูลที่เตรียมไว้รออัปเดต

4. Output

การ read คือการเข้ามาอ่านข้อมูลตัว h_t โดยจะมีการควบคุมว่าสามารถอ่านได้หรือไม่ผ่าน sigmoid function กับค่า hidden state ตัวก่อนหน้า กับ input data ที่เข้ามาโดยสมการจะคล้ายกับ input gate และ forget gate ที่ผ่านมา

$$o_t = \sigma(x_t U^o + h_{t-1} W^o) \quad \text{.....(10)}$$

σ คือ sigmoid function
x_t คือค่าอินพุทขาเข้า
h_{t-1} คือค่า hidden state ก่อนหน้า
U^o คือค่า weight ของการ output gate
W^o คือค่า weight ของ hidden state
b^i คือค่า bias ของการ output gate

5. hidden state

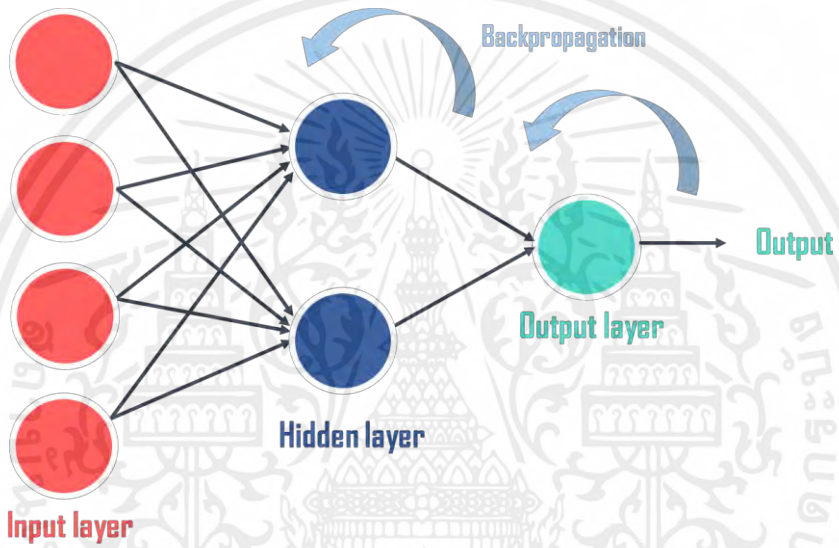
โดยค่าจะเราจะส่ง output เป็นค่า h_t สำหรับ sequence ถัดไป และ layer ถัดไปจะรวมเป็นสมการของ hidden state ออกมาดังนี้

$$h_t = tanh(C_t) * o_t \quad \text{.....(11)}$$

ในส่วนของสมการในการส่งค่า output
โดย o_t มีค่าเป็น 0 จะไม่ส่งค่า output ออกไปและค่าของ h_t จะมีค่าเท่ากับ 0
o_t มีค่าเป็น 1 จะส่งค่า output ออกไปและค่าของ h_t จะสามารถอ่านได้

2.5 Backpropagation

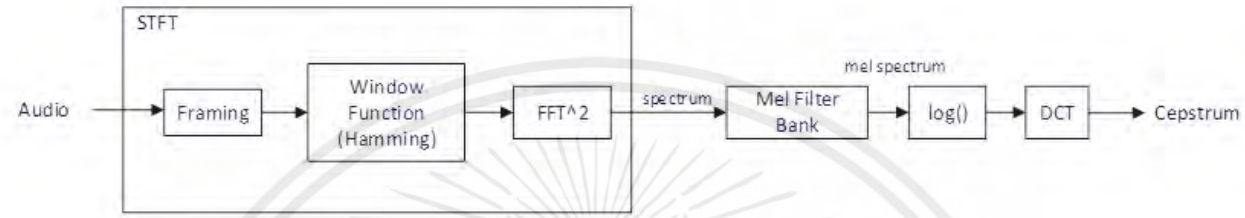
จากแต่ละชั้นใน neural network จะเห็นถึงการปรับพารามิเตอร์ที่สำคัญ 2 คือ weight และ bias โดยในการ train โมเดลนั้น algorithm ที่เรานิยมใช้กันก็คือ Backpropagation โดยจะทำการป้อนข้อมูลที่มีคำตอบที่รู้แน่นอนใส่เข้าไปในโมเดลและคำนวณค่า error จาก output layer เทียบกับคำตอบที่ควรจะเป็นซึ่ง Backpropagation จะนำ error ที่ได้มาเพื่อใช้ปรับ weight และ bias ของโมเดลไปเรื่อยๆจนกว่า output layer จะมี error ต่ำที่สุด



รูปที่ 2.5.1 Backpropagation

2.6 Mel Frequency Cepstrum coefficient (MFCC)

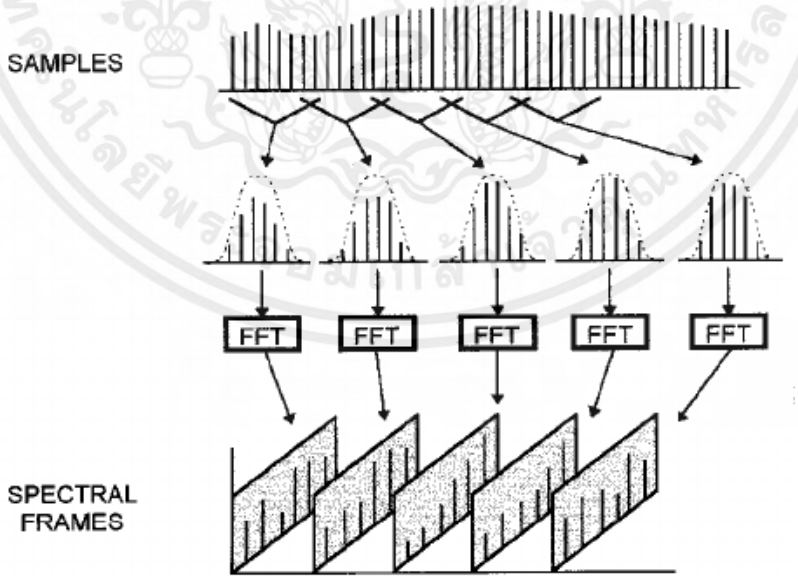
MFCC เป็นการสกัดลักษณะเด่น เป็นการดึงลักษณะเฉพาะของหน่วยเสียงแต่ละหน่วยเสียง ที่แตกต่างกันออกมา แล้วให้ระบบสามารถจดจำลักษณะเด่นของแต่ละเสียงไว้ ข้อมูลจำนวนมากของข้อมูลเสียงจะถูกแปลงเป็น coefficient จะถูกแปลงเป็นชุดข้อมูลที่มีจำนวนน้อยลง และยังคงคุณสมบัติสำคัญของข้อมูลเดิมไว้ได้อย่างถูกต้อง โดยมีหลักการดังต่อไปนี้



รูปที่ 2.6.1 ขั้นตอนการทำ MFCC

1. Short-Time Fourier Transform (STFT)

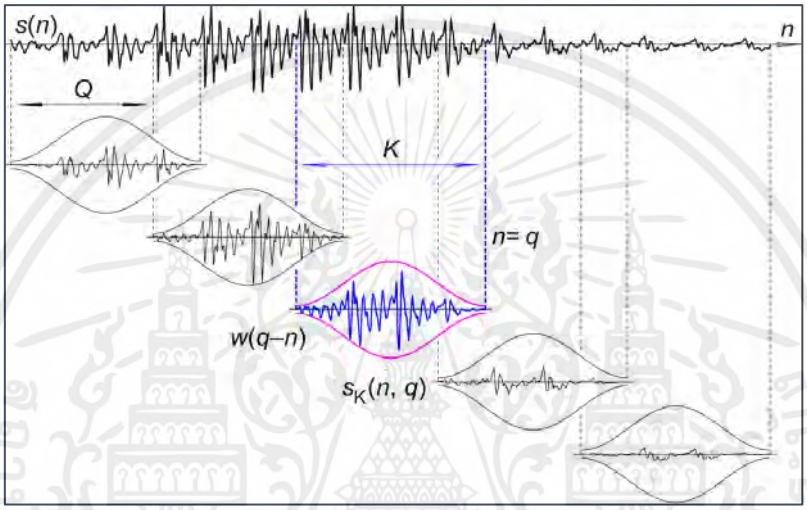
เป็นการแบ่งสัญญาณเสียงได้จากการ Sampling ออกเป็นเฟรม และผ่าน Window Function เช่น Hamming เพื่อปรับค่าของสัญญาณลงในบางจุดให้ลดลง แล้วสุดท้ายจึงทยอยทำ FFT ที่ละเฟรมของเสียงออกมาเป็น spectrum ของสัญญาณเสียง



รูปที่ 2.6.2 ขั้นตอนการทำ STFT

Framing

Framing คือการแบ่งเฟรมของข้อมูลเสียงออกเป็นช่วงๆแต่ละเฟรมโดยในการทดลองเราจะแบ่งข้อมูลเพื่อสกัดลักษณะเด่นใน mfcc ทีละ 2048 ข้อมูลต่อ 1 เฟรม และมี hop length ที่เป็นข้อมูลเหลื่อมทับกันในแต่ละเฟรมขนาด 512 ข้อมูลโดยตัวอย่างสัญญาณ sampling rate ที่เก็บข้อมูลมาคือ 44100 จะแบ่งเฟรมที่ต้องทำทั้งหมดเป็น 22 เฟรมจากข้อมูล 1 วินาที



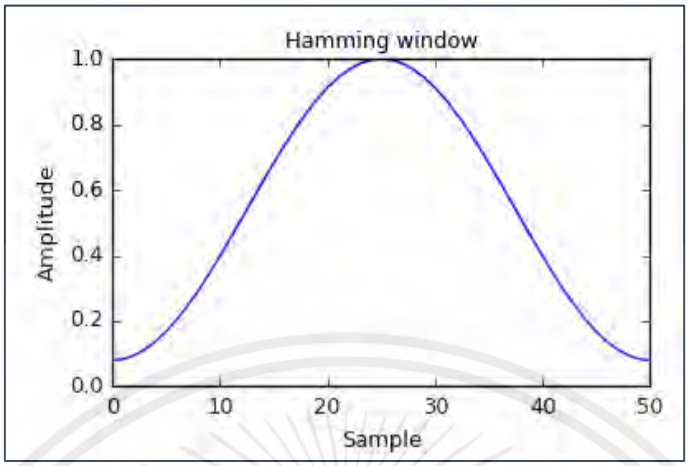
รูปที่ 2.6.3 framing

Window function (Hamming)

Window Functions คือ Functions ที่ทำการคำนวณผ่าน Rows ต่าง ๆ ของ Dataset โดยที่ยังคงรักษา Rows และจำนวน Rows ใน Original Table ให้อยู่คงเดิม โดยการใช้แบบ Hamming ทำเพื่อลดความแตกต่างระหว่างข้อมูลแต่ละเฟรมฟังก์ชันนี้เป็นสมาชิกของทั้งตระกูลผลรวมโคไซน์และตระกูลกำลังของไซน์ ต่างจากหน้าต่าง Hamming จุดสิ้นสุดของหน้าต่าง Hann เพียงแตะศูนย์ ผลที่ได้จะออกที่ประมาณ 18 เดซิเบลต่ออ็อกเทฟ เขียนเป็นสมการได้ดังนี้

$$w[n] = 0.5\,[1 - \cos(\frac{2\pi n}{N})] \quad \text{.....(12)}$$

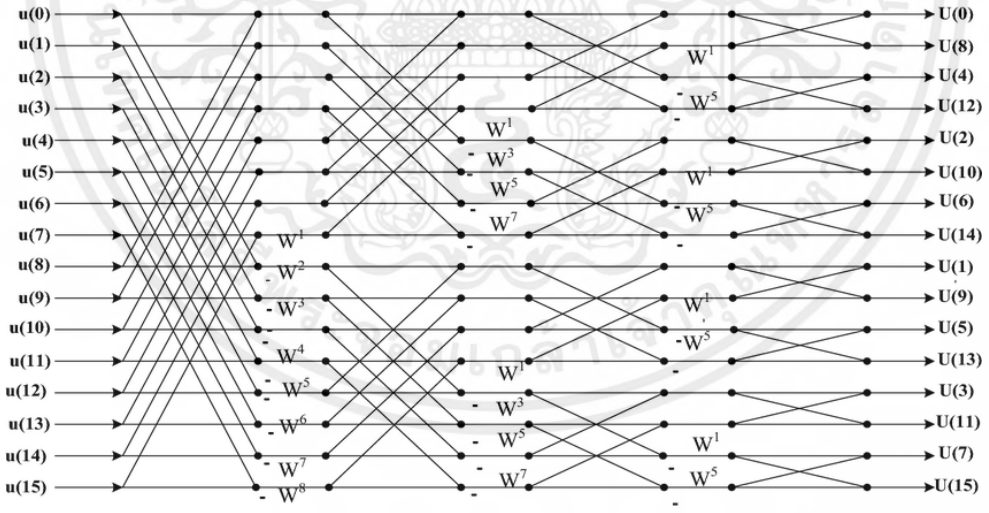
กราฟจากสมการ Window function (Hamming)



รูปที่ 2.6.4 Window function (Hamming)

Fast Fourier Transform

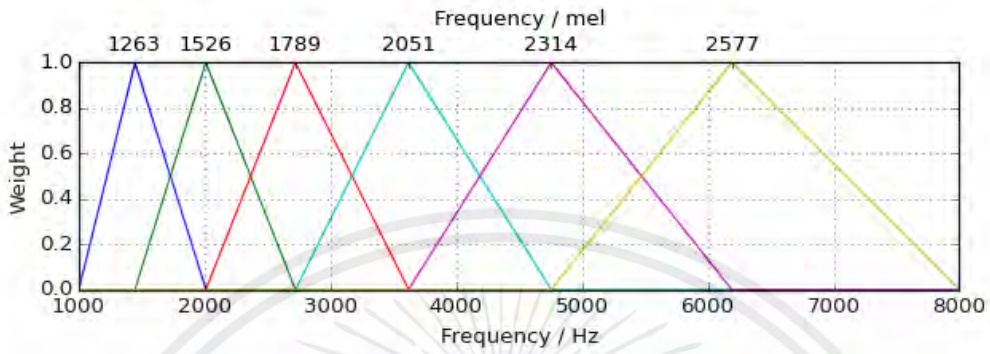
การทำ FFT หรือ Fast Fourier Transform คือการแปลงข้อมูลจากแกนเวลาเป็นแกนความถี่โดยในโปรเจคนี้จะใช้ FFT Butterfly โดยในโปรเจคเราจะใช้ 2048 point แสดงเป็นแผนผังเส้นได้ดังตัวอย่าง



รูปที่ 2.6.5 ตัวอย่าง FFT Butterfly 16 point

2. Mel Filter Bank

นำสัญญาณที่ได้จากการทำ STFT มาผ่าน Mel Filter Bank เพื่อกรองสัญญาณอื่นๆที่ไม่ใช่ลักษณะของเสียงออกและเน้นความสำคัญของความถี่ที่อยู่ในช่วงกลางของ Mel Filter Bank มีลักษณะดังนี้



รูปที่ 2.6.6 Mel Filter Bank

เมื่อ spectrum ของสัญญาณเสียง ผ่าน Mel Filter Bank จะกลายเป็น Mel spectrum

3. Discrete Cosine Transform (DCT)

หลังจากนำ Mel spectrum มาใส่ log() แล้ว เพื่อเปลี่ยนสัญญาณกลับไปเป็นในโดเมนของเวลาจึงต่องใช้การ DCT เพื่อแปลง Mel spectrum ในโดเมนความถี่กลับมาเป็นในโดมเมนเวลาโดยในโปรเจคนี้เราจะใช้ Discrete Cosine Transform แบบ type ii ในการคำนวณ

สมการของ DCT type II

$$X_k = \sum_{n=0}^{N-1} x_n cos\left[\frac{\pi}{N}(n+\frac{1}{2})k\right] \quad for\ k = 0, \dots N-1 \quad \text{.....(13)}$$

บทที่ 3
ขั้นตอนการดำเนินงาน

3.1 การจัดการข้อมูล

จากการเขียนโปรแกรมส่วนการทำงานของระบบบนชิปสำหรับการประมวลผลและรู้จำเสียงพูดในเบื้องต้นทดสอบบน python โดยใช้คำสั่งไลบารีของ python 3.11.3 มาใช้ในการจัดการข้อมูล และสร้างส่วนประมวลผลโครงข่ายประสาทเทียมเพื่อวิเคราะห์ และจัดการปรับปรุงข้อมูลให้เหมาะสม โดยสามารถเรียน รู้ที่จะทำงานที่มอบหมายได้ โดยใช้เวลาประมวลผลน้อยและมีระบบมีความแม่นยำ ในการแยกแยะได้

3.1.1 ชุดข้อมูล (data set)

ในโครงงานนี้เราต้องการใช้ชุดข้อมูลเป็นคำพูดภาษาไทย 4 คำคือคำว่า“ไป,หยุด,ซ้าย,ขวา”ในการเป็นเงื่อนไขให้ตัวระบบสามารถแยกแยะคำพูด 4 คำนี้ในลักษณะเสียงที่มีความแตกต่างกันให้สามารถระบุคำตอบของเสียงนั้นออกมาได้ โดยชุดข้อมูลทั้งหมดที่ทำการทดสอบคือ 48 เสียงต่อ 1 คำ รวมกันเป็น 192 ชุด โดยชุดข้อมูลโดยสามารถแบ่งออกเป็นข้อมูลที่ใช้ทำการเรียนรู้สอนและทำการทดสอบได้ดังนี้

ข้อมูลสำหรับทำการสอน (Train data)

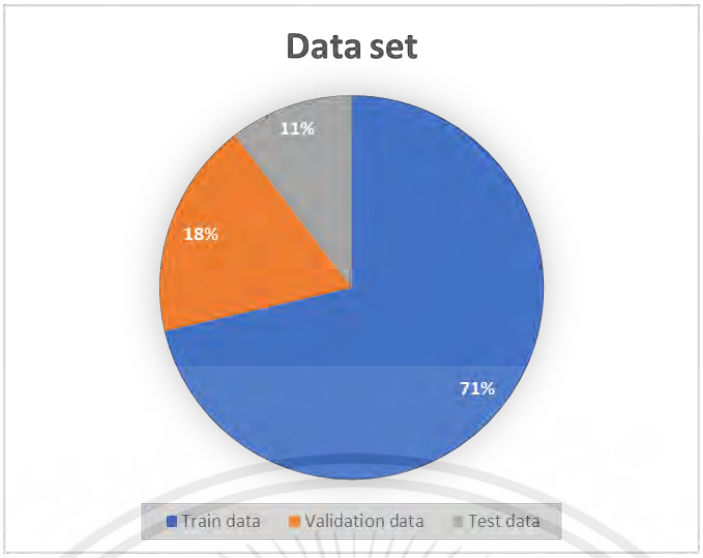
เป็นข้อมูลที่ใช้ในการสอนตัวโมเดลระบบของเราให้มีความสามารถในการแบ่งแยกและจดจำคุณลักษณะของเสียงในคำเฉพาะต่างๆ

ข้อมูลสำหรับทำการตรวจสอบ (Validation data)

เป็นข้อมูลที่ใช้ในการตรวจสอบหลังจากที่ทำการสอนว่าระบบของเราไม่เคยใช้ข้อมูลชุดนี้จะ สามารถแยกแยะทำนายได้มีความแม่นยำแค่ไหนและใช้เพื่อปรับจูน weight กับ bias ในระบบของเรา

ข้อมูลสำหรับทำการทดสอบ (Test data)

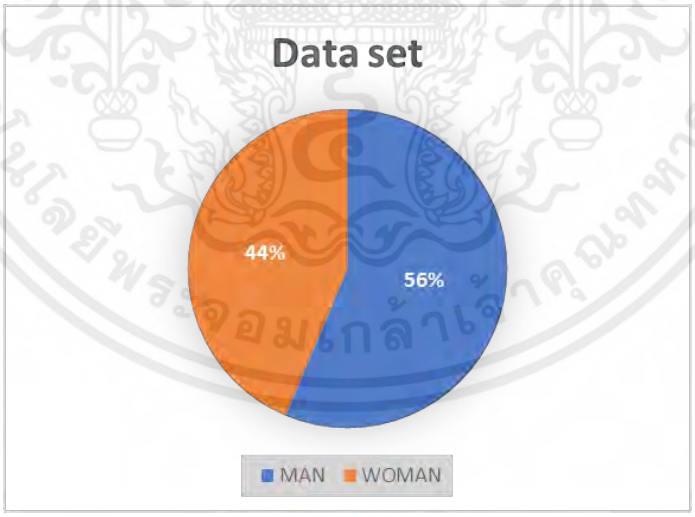
เป็นข้อมูลที่ใช้สำหรับทำการทดสอบโดยจะไม่เคยใช้ข้อมูลชุดนี้ในการเรียนรู้ของระบบ แต่ใช้ไว้ตรวจสอบความถูกต้องว่าสามารถแยกแยะคำตอบได้ถูกต้องหรือไม่



รูปที่ 3.1.1 กราฟแผนภูมิการแบ่งสัดส่วนข้อมูล

สัดส่วนการใช้ชุดข้อมูล 192 ชุด (1 คำ 48ชุด)

ข้อมูลสำหรับทำการสอน	Train data	137 sample
ข้อมูลสำหรับทำการตรวจสอบ	Validation data	35 sample
ข้อมูลสำหรับทำการทดสอบ	Test data	20 sample



รูปที่ 3.1.2 กราฟแบ่งสัดส่วนข้อมูลชายและหญิง

สัดส่วนการใช้ชุดข้อมูล 192 ชุด (1 คำ 48ชุด)

เสียงผู้ชาย	124 sample
เสียงผู้หญิง	68 sample

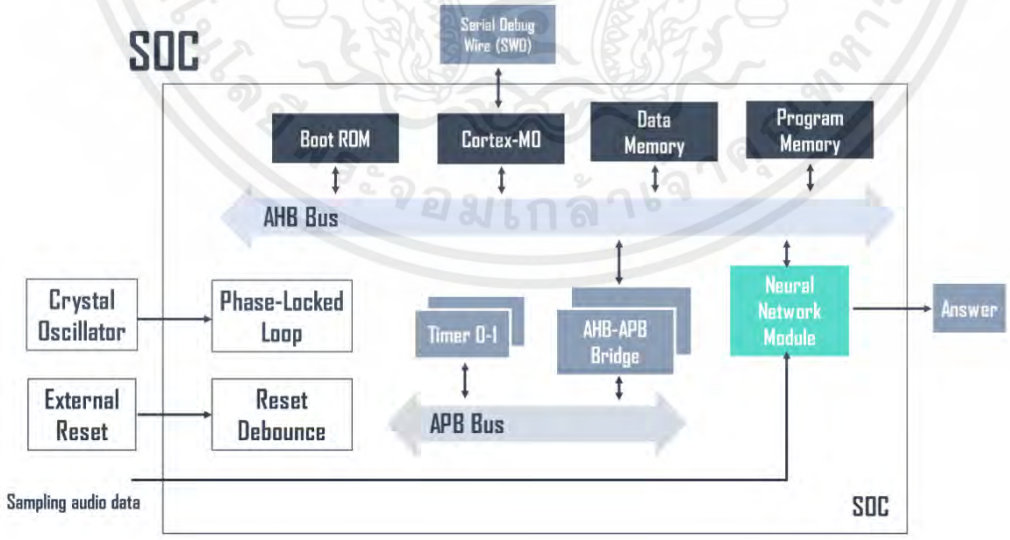
3.2 การวิเคราะห์ข้อมูล

การทดสอบการลดความละเอียดของระบบ LSTM (Long Short Term Memory) โดยหลังจากที่ ได้ทำการใช้ระบบ LSTM (128) ต่อกับ LSTM2 (64) และต่อกับ DENSE (64) พบว่าการทำงานของระบบ มีความแม่นยำแต่ใช้จำนวนข้อมูลและการทำงานที่นานจึงได้ทำการ ลดความละเอียดของ LSTM แบ่งออกได้เป็น 4 ระบบดังนี้

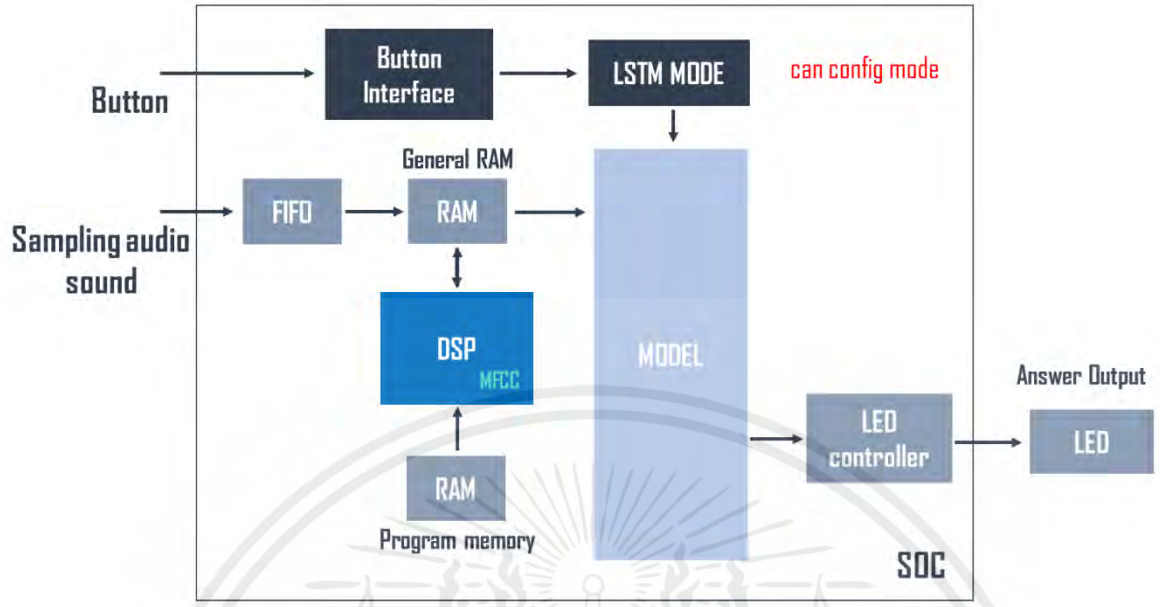
- LSTM 128 --> LSTM2 64 --> DENSE 64
- LSTM 64 --> LSTM2 32 --> DENSE 32
- LSTM 32 --> LSTM2 16 --> DENSE 16
- LSTM 16 --> LSTM2 8 --> DENSE 8

โดย LSTM ที่มีขนาดเล็กลงจะลดพารามิเตอร์ในการฝึกและการคำนวณทำให้สามารถประมวลผลด้วยความเร็วที่ดีกว่าแต่ชดเชยมาด้วยความละเอียดในการตรวจสอบแม่นยำของระบบที่ลดลง

3.3 แผนผังบล็อกของระบบบนชิปประมวลผลและรู้จำเสียงพูด



รูปที่ 3.4.1 แผนผังบล็อคของระบบบนชิปทั้งหมด



รูปที่ 3.4.2 แผนผังบล็อคของระบบบนชิปในส่วนของระบบประมวลผลสัญญาณดิจิทัล

3.4 การออกแบบหน่วยประมวลผลสัญญาณดิจิทัล (cmsdk_ecg_dsp)

ในการออกแบบหน่วยประมวลผลสัญญาณดิจิทัล จะต้องประกอบไปด้วยโมดูลย่อยๆสาม โมดูลนั่นคือ Fetch, Decode และ Execute ซึ่งในแต่ละโมดูลจะมีการทำงานที่เหมือนกับการทำงาน ในหน่วยประมวลผลทั่วไป โดยจะอธิบายภายรวมของการทำงานได้ดังนี้ ในตอนเริ่มต้น หน่วย ประมวลผลสัญญาณดิจิทัลจะอยู่ในสถานะหยุด (Halt) จนกว่าจะมีการสั่งให้เริ่มทำงาน และเลือกอ่าน ที่ตำแหน่งของ Program Memory ด้วย Cortex-M0 ต่อไป

3.4.1 การออกแบบหน่วยอ่านคำสั่ง (cmsdk_dsp_fetch)

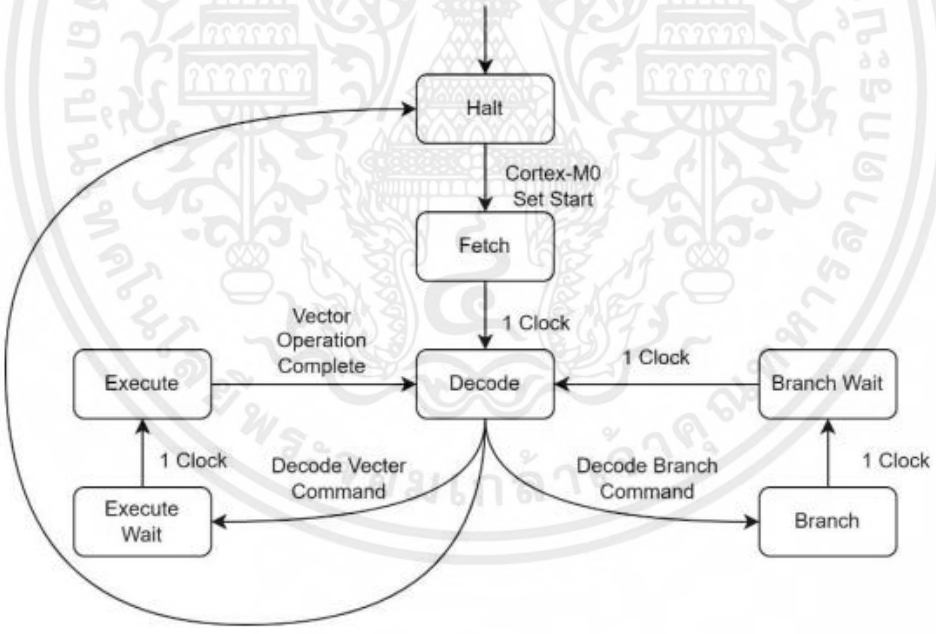
ในการออกแบบหน่วยดึงข้อมูล เมื่อเริ่มต้นจะไม่ทำการอ่านคำสั่งใดๆจากหน่วยความจำเลย จนกว่า Cortex-M0 จะสั่งให้เริ่มดำเนินการ และ Cortex-M0 สามารถตั้งค่าตำแหน่งการอ่านของ หน่วยความจำโปรแกรมได้จากการตั้งค่าผ่านรีจิสเตอร์ แต่จะไม่สามารถตั้งค่าตำแหน่งได้ด้วยตัวเอง เว้นแต่การกระโดดข้ามโดยใช้คำสั่ง BNE และ BEQ ที่จะกล่าวถึงในส่วนถัดไป

3.4.2 การออกแบบหน่วยตีความคำสั่ง (cmsdk_dsp_decode)

หน่วยตีความคำสั่ง จะเริ่มต้นทำงานในสถานะหยุด (Halt) และจะเริ่มตีความคำสั่งเมื่อมีการ อ่านคำสั่งแล้วซึ่งก็คือหลังจากที่ Cortex-M0 ส่งคำสั่งมา 1 คาบนาฬิกาและหน่วยนี้จะคอยควบคุมการ อ่าน/เขียนระหว่างแรมทั่วไป (General RAM) และการเขียนเข้า LCD FIFO ทั้งหมดรวมทั้งสถานะ เครื่องทุกสถานะหน่วยนี้จะเป็นผู้ควบคุมแต่เพียงผู้เดียว

3.4.3 การออกแบบหน่วยดำเนินการคำสั่ง (cmsdk_dsp_execute)

หน่วยดำเนินการ จะเป็นหน่วยที่ภายในจะประกอบไปด้วย Intellectual Property ของ Xilinx ที่เป็นส่วนของการดำเนินการทางตัวเลขจุดลอยตัว (Floating-Point Operation) ทั้งหมด ซึ่ง หน่วยนี้จะประกอบไปด้วยมัลติเพล็กเซอร์จำนวนมากเพื่อเลือกช่องทางการป้อนข้อมูลต่างๆให้กับ หน่วยประมวลผล



รูปที่ 3.4.1 แผนผังสถานะของหน่วยประมวลผลสัญญาณดิจิทัล

3.4.4 การออกแบบสถาปัตยกรรมชุดคำสั่งเครื่อง (Instruction Set Architecture Design)

คำสั่งเครื่องที่ใช้ในหน่วยประมวลผลดิจิทัลนี้ จะประกอบไปด้วยชุดคำสั่งพื้นฐานที่สามารถ พบได้ทั่วไป และชุดคำสั่งพิเศษที่สร้างขึ้นมาในการประมวลผลสัญญาณดิจิทัลโดยเฉพาะ สามารถ จำแนกเป็นรายการได้ดังนี้

ตารางที่ 1 คำสั่งที่มีในหน่วยประมวลผลสัญญาณดิจิทัล

คำสั่ง	รายละเอียด	ตัวดำเนินการ	Program Counter
HALT	ไม่ดำเนินการใดๆ		PC + 0
MOV	โหลดข้อมูลในชุดคำสั่ง	Rd ⟵ Data	PC + 1
ADD	บวกค่าของรีจิสเตอร์ที่เลือกด้วยข้อมูลในชุดคำสั่ง	Rd ⟵ Rd + Data	PC + 1
SUB	ลบค่าของรีจิสเตอร์ที่เลือกด้วยข้อมูลในชุดคำสั่ง	Rd ⟵ Rd - Data	PC + 1
AND	ดำเนินการและทางบิต ของรีจิสเตอร์ที่เลือกกับ ข้อมูลในชุดคำสั่ง	Rd ⟵ Rd & Data	PC + 1
OR	ดำเนินการหรือทางบิต ของรีจิสเตอร์ที่เลือกกับ ข้อมูลในชุดคำสั่ง	Rd ⟵ Rd | Data	PC + 1
NOT	ดำเนินการนิเสธทางบิต ของรีจิสเตอร์ที่เลือกกับ ข้อมูลในชุดคำสั่ง	Rd ⟵ ~Rd	PC + 1
XOR	ดำเนินการและเฉพาะ ทางบิตของรีจิสเตอร์ที่ เลือกกับข้อมูลในชุดคำสั่ง	Rd ⟵ Rd $\oplus$ Data	PC + 1
LDM	อ่านข้อมูลจากแรมทั่วไป ที่ตำแหน่งของรีจิสเตอร์ ตำแหน่งข้อมูลขาเข้า แล้วเก็บค่าไว้ที่รีจิสเตอร์ อ่านข้อมูล		PC + 1
STO	เขียนข้อมูลลงแรมทั่วไป ที่ตำแหน่งของรีจิสเตอร์ ตำแหน่งข้อมูลขาออกด้วยข้อมูลที่อยู่ภายใน รีจิสเตอร์เขียนข้อมูล		PC + 1
BNE	ตรวจสอบข้อมูลที่ รีจิสเตอร์ที่เลือกกับ รีจิสเตอร์เปรียบเทียบ		ถ้าไม่ตรง PC + k ถ้าตรง PC + 1

BEQ	ตรวจสอบข้อมูลที่ รีจิสเตอร์ที่เลือก กับ รีจิสเตอร์เปรียบเทียบ		ถ้าไม่ตรง PC + 1 ถ้าตรง PC + k
VECT	ดำเนินการทางเวกเตอร์ ด้วยคำสั่งภายในชุด ข้อมูลที่กำหนด		

โดย 1 ชุดคำสั่ง (ยกเว้นคำสั่ง VECT) จะมีองค์ประกอบทางบิตดังต่อไปนี้

ตารางที่ 2 องค์ประกอบทางบิตของชุดคำสั่งทั่วไป

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
รหัสคำสั่ง					รีจิสเตอร์			ข้อมูล															

และสำหรับชุดคำสั่งของ VECT จะมีองค์ประกอบทางบิตดังต่อไปนี้

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VECT					คำสั่งเวกเตอร์ทั่วไป					คำสั่งเฉพาะ			RESERVED										

3.4.5 การออกแบบชุดคำสั่งทางเวกเตอร์ (Vector Operation Design)

ในการออกแบบคำสั่งทางเวกเตอร์ จะประกอบด้วยคำสั่งทางเวกเตอร์ทั่วไป และคำสั่งทาง เวกเตอร์เฉพาะ ซึ่งคำสั่งทางเวกเตอร์ทั่วไปจะเป็นคำสั่งที่ใช้บอกการดำเนินการทางคณิตศาสตร์มี ดังต่อไปนี้

1) ADD - การบวก

2) SUB - การลบ

3) MUL - การคูณ

4) DIV – การหาร

5) EXP - การยกกำลังด้วยฐานธรรมชาติ

6) LOG – การทำลอการิทึมฐานธรรมชาติ

7) CMP – การเปรียบเทียบ

8) X2F – การแปลงเลขจุดคงที่เป็นเลขจุดลอยตัว

9) F2X – การแปลงเลขจุดลอยตัวเป็นจุดคงที่

10) FIR – การดำเนินการทางฟิลเตอร์แบบผลตอบสนองต่ออิมพัลส์จำกัด

11) USP – การเพิ่มอัตราการสุ่มด้วยการเติม 0 (Upsample)

12) DSP - การลดอัตราการสุ่ม (Downsample)

และ คำสั่งทางเวกเตอร์เฉพาะจะเป็นการบอกลักษณะของการดำเนินการโดยสามารถแบ่งได้ดังนี้

1) การดำเนินการแบบสมาชิกต่อสมาชิก (Element-Wise Operation) เช่น การคูณกับแบบสมาชิกต่อสมาชิกของเวกเตอร์

2) การดำเนินการแบบสะสม (Cumulative Operation) เช่น การหาผลบวกทั้งหมดของเวกเตอร์

3) การดำเนินการแบบค่าคงที่ (Constant Operation) เช่น การบวกทั้งเวกเตอร์ด้วยค่าคงที่

4) การดำเนินการแบบตัวแปรเดียว (Single Variable Operation) เช่น การแปลงตัวเลขจุดคงที่เป็นตัวเลขจุดลอยตัว

5) การดำเนินการแบบหาค่า (Find Operation) เช่น การหาสูงสุดในเวกเตอร์

6) การดำเนินการแบบ FIR (FIR Operation) คือการดำเนินการประมวลผลของสัญญาณเมื่อนำไปผ่านฟิลเตอร์ชนิดผลตอบสนองต่ออิม พัลส์จำกัด โดยในแต่ละคำสั่งเวกเตอร์ทั่วไป จะสามารถเลือกคำสั่งเวกเตอร์เฉพาะได้เพียงบางคำสั่งเท่านั้น สามารถสรุปได้ดังตารางด้านล่าง

ตารางที่ 3 การใช้งานคำสั่งเวกเตอร์ทั่วไป กับ คำสั่งเวกเตอร์เฉพาะ

คำสั่งเวกเตอร์ทั่วไป	Element Wise	Cumulative	Constant	FIR	Single Variable
ADD					
SUB					
MUL					
DIV					
EXP					
LOG					
CMP					
X2F					
F2X					
FIR					
USP					
DSP					

3.4.6 การออกแบบรีจิสเตอร์ภายในหน่วยประมวลผลดิจิทัล (Internal DSP Register Design)

ในการใช้งานหน่วยประมวลผลดิจิทัล จำเป็นที่จะต้องมีการตั้งค่ารีจิสเตอร์ต่างๆ เพื่อให้ สามารถทำงานได้อย่างถูกต้อง โดยจะมีรีจิสเตอร์พื้นฐานดังแสดง

ตารางที่ 4 รีจิสเตอร์ภายในของหน่วยประมวลผลสัญญาณดิจิทัล

ชื่อรีจิสเตอร์	รายละเอียด	ขนาด
Input Address	ตำแหน่งของข้อมูลขาเข้าที่ 1	16B
Coefficient Address	ตำแหน่งของข้อมูลขาเข้าที่ 2	16B
Output Address	ตำแหน่งของข้อมูลขาออก	16B
Size	ขนาดของเวกเตอร์ที่จะดำเนินการ	16B
Tap	จำนวนของสัมประสิทธิ์สำหรับการดำเนินการทางฟิลเตอร์ แบบผลตอบสนองต่ออิมพัลส์ จำกัด	16B
General	การตั้งค่าการดำเนินการทั่วไป	16B
Compare	รีจิสเตอร์เปรียบเทียบ ใช้กับคำสั่ง BNE หรือ BEQ	16B
AddrAlt	รีจิสเตอร์ตั้งค่าการเลือก increment ของ ตำแหน่งข้อมูลขาเข้า1,2 และ เอาท์พุต	16B
AddrFlip	รีจิสเตอร์ตั้งค่าให้ตำแหน่งของข้อมูลขาเข้า2 กลับบิต	16B
Gen Ram Write Data H	รีจิสเตอร์เก็บข้อมูลที่จะเขียน ลงแรมทั่วไปในส่วนของบิตสูง (บิตที่ 31 – 16)	16B
Gen Ram Write Data L	รีจิสเตอร์เก็บข้อมูลที่จะเขียน ลงแรมทั่วไปในส่วนของบิตสูง (บิตที่ 15 – 0)	16B
Gen Ram Read Data	รีจิสเตอร์เก็บข้อมูลที่อ่านจากแรมทั่วไป	32B

ตารางที่ 5 องค์ประกอบทางบิตของรีจิสเตอร์ rAddrAlt

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
shield guard				Inc Output				Inc Intput2				Inc input1			

Bit 15-12 : shield guard จะอ่านตัวเลขจากบิตนี้ไปใช้ในการ guard ข้อมูลขาเข้าที่ 2 โดยการ AND เพื่อให้อ่านข้อมูลเป็นลำดับในการทำ FFT Butterfly เช่น บิตที่ 15-12 มีค่าเท่ากับ 0010 คือมีค่าเท่ากับ 2 ในเลขฐานสิบ แสดงว่าต้องการ guard ทั้งหมด 2 บิตคือ 11100 โดยจากตัวอย่างตำแหน่งของข้อมูลขาเข้า2 จาก 00110 คือค่า 6 ในเลขฐานสิบ จะกลายเป็น 00110 & 11100 = 00100 คือ 4 ในเลขฐานสิบแทนตำแหน่งเดิมที่มีค่า 6

Bit 11-8 : Inc Output คือการกำหนดค่า Increment ที่คอยชี้ตำแหน่งที่คอยบวกเพิ่มจากเดิมทีละ 1 ของข้อมูลขาออกให้เป็นบวกเพิ่มทีละจำนวนตามข้อมูลในบิตที่ 11-8 เช่น บิตที่ 11-8 มีค่าเท่ากับ 0011 คือมีค่าเท่ากับ 3 ในเลขฐาน สิบ แสดงว่าต้องการกำหนดค่า Increment ที่คอยชี้ตำแหน่งของข้อมูลขาออก ให้บวกเพิ่มทีละ 3 ตำแหน่ง

Bit 7-4 : Inc Input2 คือการกำหนดค่า Increment ที่คอยชี้ตำแหน่งที่คอยบวกเพิ่มจากเดิมทีละ 1 ของข้อมูลขาเข้า2 ให้เป็นบวกเพิ่มทีละจำนวนตามข้อมูลในบิตที่ 7-4 เช่น บิตที่ 7-4 มีค่าเท่ากับ 0010 คือมีค่าเท่ากับ 2 ในเลขฐาน สิบ แสดงว่าต้องการกำหนดค่า Increment ที่คอยชี้ตำแหน่งของข้อมูลขาเข้า2 ให้บวกเพิ่มทีละ 2 ตำแหน่ง

Bit 3-0 : Inc Intput1 คือการกำหนดค่า Increment ที่คอยชี้ตำแหน่งที่คอยบวกเพิ่มจากเดิมทีละ 1 ของข้อมูลขาเข้า1 ให้เป็นบวกเพิ่มทีละจำนวนตามข้อมูลในบิตที่ 3-0 เช่น บิตที่ 3-0 มีค่าเท่ากับ 0100 คือมีค่าเท่ากับ 4 ในเลขฐานสิบ แสดงว่าต้องการกำหนดค่า Increment ที่คอยชี้ตำแหน่งของข้อมูลขาเข้า1 ให้บวกเพิ่มทีละ 4 ตำแหน่ง

ตารางที่ 6 องค์ประกอบทางบิตของรีจิสเตอร์ rAddrFlip

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Flip constant															

Bit 15-0 : Flip constant คือการกำหนดค่าจำนวนที่เราต้องการ flip บิตกลับหัวท้ายของข้อมูลขาเข้า2 โดยบิตที่ 15-0 จะเป็นการกำหนดค่าจำนวนบิตที่เราต้องการกลับบิต เช่น เราให้ตำแหน่งข้อมูลขาเข้า2 มีค่า 00101011 คือมีค่าเท่ากับ 43 ในเลขฐานสิบ แล้วกำหนด rAddrFlip บิตที่ 15-0 มีค่า 00100 ซึ่งมีค่าเท่ากับ 4 ในเลขฐานสิบ จะกลับบิตของข้อมูลขาเข้า 2 ทั้งหมด 4 บิต คือ 00101011 --> 00101101

3.4.7 การออกแบบภาษาและหน่วยตีความสำหรับหน่วยประมวลผลดิจิทัล (Language and Interpreter Design for DSP)

ในการออกแบบภาษาเพื่อใช้สำหรับเขียนโปรแกรมให้กับหน่วยประมวลผลดิจิทัล จะใช้ หลักการเขียนที่เหมือนกับภาษาแอสเซมบลี ซึ่งในการเขียนหนึ่งบรรทัด จะหมายถึง 1 คำสั่ง โดยให้แต่ละหนึ่งบรรทัด จะประกอบไปด้วยชุดข้อความ 3 ชุด โดยตัวอย่างของชุดคำสั่งทั่วไปที่ไม่ใช่คำสั่ง เวกเตอร์ และในการออกแบบหน่วยตีความ จะเขียนด้วยภาษาซี เพื่อให้ทำการอ่านไฟล์โปรแกรมที่ต้องการ และ ทำการแปลงคำสั่งในแต่ละบรรทัดให้เป็นรหัสเครื่องต่อไป

ตัวอย่างที่ 1

```
cmdMOV regIpAddr 0
cmdMOV regCoefAddr 2048
cmdMOV regOpAddr 4096
cmdMOV regSize 2048
cmdMOV regGeneral 0
cmdVECT vcmdADD Elem
```

รูปที่ 3.4.2 แสดงตัวอย่างคำสั่งบวก

```
Reading form file : .\Functional\test02.dsp...
0,
524288,
591872,
659456,
722944,
851968,
16285696,
```

รูปที่ 3.4.3 แสดงตัวอย่างผลลัพธ์การใช้งานหน่วยตีความ

ความหมายของคำสั่งในตัวอย่างที่ 1

cmdMOV regIpAddr 0	ให้ดำเนินการชี้ค่าตำแหน่งที่ 0 เป็นตำแหน่งของข้อมูลขาเข้า 1
cmdMOV regCoefAddr 2048	ให้ดำเนินการชี้ค่าตำแหน่งที่ 2048 เป็นตำแหน่งของข้อมูลขาเข้า 2
cmdMOV regOpfAddr 4096	ให้ดำเนินการชี้ค่าตำแหน่งที่ 4096 เป็นตำแหน่งของข้อมูลขาออก
cmdMOV regSize 2048	ให้ดำเนินการนำค่า 2048 ลงไปเก็บในรีจิสเตอร์ขนาด
cmdMOV regGeneral 0	ให้ดำเนินการนำค่า 0 ลงไปเก็บในรีจิสเตอร์ General
cmdVECT vcmdADD Elem	ให้ดำเนินการแบบเวกเตอร์ โดยใช้คำสั่งแบบบวก และคำสั่งทางเวกเตอร์เฉพาะให้ดำเนินการแบบสมาชิกต่อสมาชิก

ตัวอย่างที่ 2

```
cmdMOV regIpAddr 1536
cmdMOV regCoefAddr 1739
cmdMOV regOpAddr 0
cmdMOV regSize 128
cmdMOV regGeneral 0
cmdMOV regAddrFlip 11
cmdVECT vcmdMUL Cons
```

รูปที่ 3.4.4 แสดงตัวอย่างคำสั่งคูณ

```
Reading form file : .\Functional\test02.dsp...
0,
525824,
591563,
655360,
721024,
851968,
917515,
16318464,
0,
```

รูปที่ 3.4.5 แสดงตัวอย่างผลลัพธ์การใช้งานหน่วยตีความ

ความหมายของคำสั่งในตัวอย่างที่ 2

cmdMOV regIpAddr 1536 ให้ดำเนินการชี้ค่าตำแหน่งที่ 1536 เป็นตำแหน่งของข้อมูลขาเข้า 1

cmdMOV regCoefAddr 1739 ให้ดำเนินการชี้ค่าตำแหน่งที่ 1739 เป็นตำแหน่งของข้อมูลขาเข้า 2

cmdMOV regOpfAddr 0 ให้ดำเนินการชี้ค่าตำแหน่งที่ 0 เป็นตำแหน่งของข้อมูลขาออก

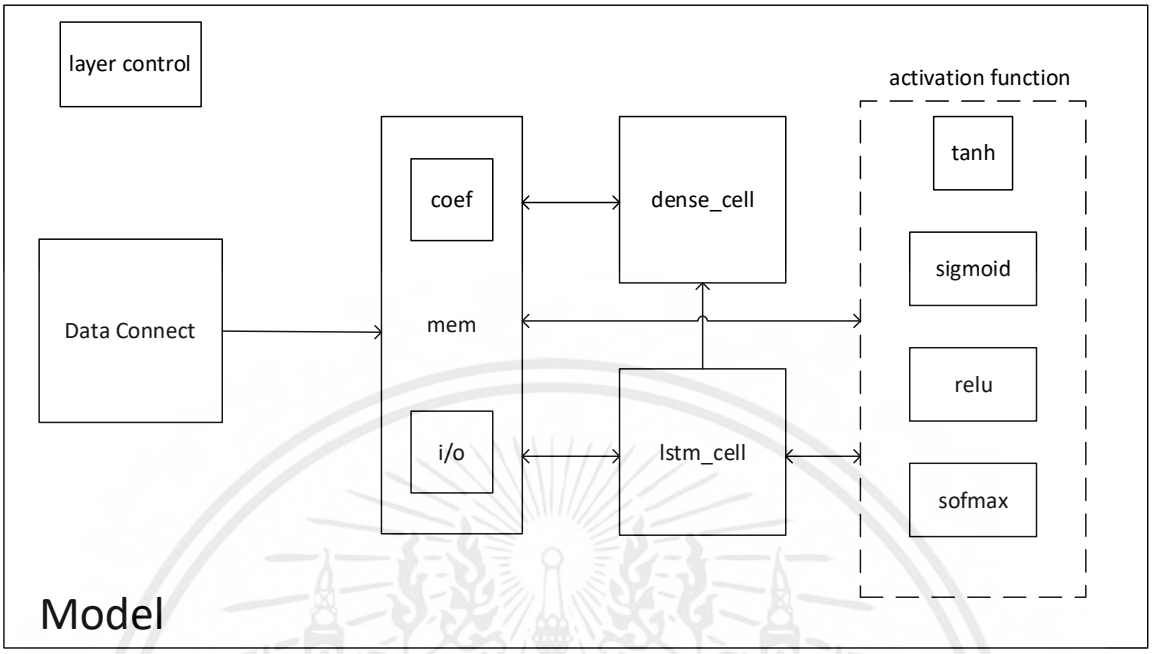
cmdMOV regSize 128 ให้ดำเนินการนำค่า 128 ลงไปเก็บในรีจิสเตอร์ขนาด

cmdMOV regGeneral 0 ให้ดำเนินการนำค่า 0 ลงไปเก็บในรีจิสเตอร์ General

cmdMOV regAddrFlip 11 ให้ดำเนินการนำค่า 11 ลงไปเก็บในรีจิสเตอร์ AddrFlip

cmdVECT vcmdMUL Cons ให้ดำเนินการแบบเวกเตอร์ โดยใช้คำสั่งแบบคูณ และคำสั่งทาง เวกเตอร์เฉพาะให้ดำเนินการให้ข้อมูลขาเข้า2 เป็นแบบค่าคงที่

3.5 การออกแบบ Model (LSTM ,Dense) เป็นวงจรดิจิตอล

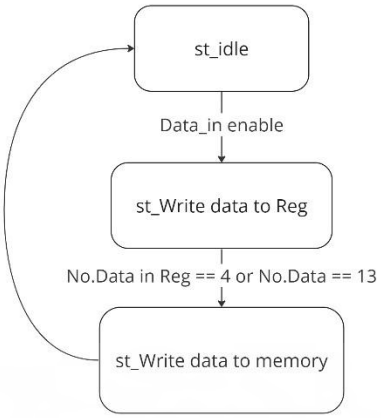


รูปที่ 3.5.1 บล็อกไดอะแกรมของวงจร model

วงจร model นั้นรับค่า mfcc coefficient จาก General Ram มาแล้วจัดเรียงข้อมูลเพื่อเขียนลงใน i/o mem ผ่านบล็อก Data Connect แล้วจึงเริ่มการทำงานของวงจร model อย่างแท้จริง โดยจะสั่งใช้งานวงจร lstm และ dense เพื่อนำ mfcc coefficient มาประมาลผลออกมาเป็นค่าความเป็นไปได้ของคำตอบแล้วส่งค่าคำตอบที่มีความเป็นไปได้มากที่สุดเพื่อส่งออกไป

3.5.1 การออกแบบวงจร Data Connect

วงจร Data Connect เมื่อมีค่า mfcc coefficient ถูกส่งเข้ามาจะนำมาเก็บในรูป Half precision (16 บิต) ภายใน register ขนาด 64 บิต พอนำ mfcc coefficient มาเก็บครบทั้งหมด 64 บิต ก็จะเขียนลงไปใน i/o mem หรืออีกกรณีนึงที่ทำให้วงจรนี้เขียนข้อมูลลงไปใน i/o mem คือ counter ภายในสามารถนับจำนวน mfcc coefficient ที่เข้ามาได้ทั้งหมด 13 ค่าซึ่งเป็นจำนวนสูงสุดของ input ของวงจรนี้จึงไม่ต้องสนใจว่า register ขนาด 64 บิต จะเขียนข้อมูลครบหรือไม่ แล้วสามารถเขียนข้อมูลลงใน i/o mem ได้เลยทันที



รูปที่ 3.5.2 Stage การทำงานของวงจร Data Connect

3.5.2 การจัดการข้อมูลของ mem i/o และ mem coef

mem i/o เป็นหน่วยความจำไว้ใช้ในการเก็บค่าของ input และ output ของแต่ละวงจรภายใน Model ซึ่งสามารถอ่านและเขียนข้อมูลลงในหน่วยความจำนี้ได้

mem coef เป็นหน่วยความจำไว้ใช้เก็บค่า weight ,bias และค่าคงที่ของ activation function ที่ไว้ใช้ในการคำนวณ ซึ่งจะสามารถอ่านค่าออกมาจากหน่วยความจำนี้ได้อย่างเดียวเท่านั้น

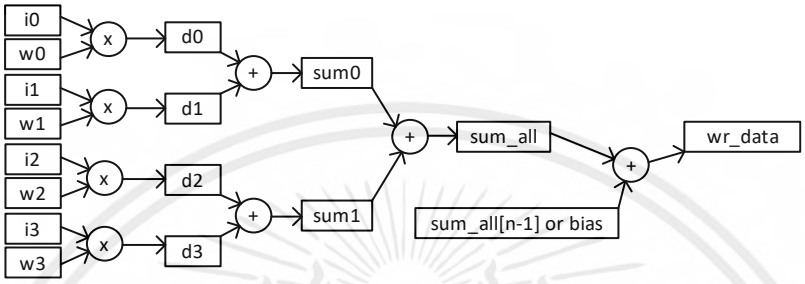
ทั้งนี้หน่วยความจำทั้งสองก็มีรูปแบบการเก็บของมูลแบบเดียวกันคือเป็นหน่วยความจำขนาด 64 บิต และภายใน 64 บิต นั้นจะแบ่งออกเป็นข้อมูล 4 ชุดเก็บในแบบ Half precision(16 บิต) ดังนี้

ตารางที่ 7 องค์ประกอบทางบิตของ mem i/o และ mem coef

63	...	...	48	47	...	...	32	31	...	...	16	15	...	...	0
Data0				Data1				Data2				Data3			

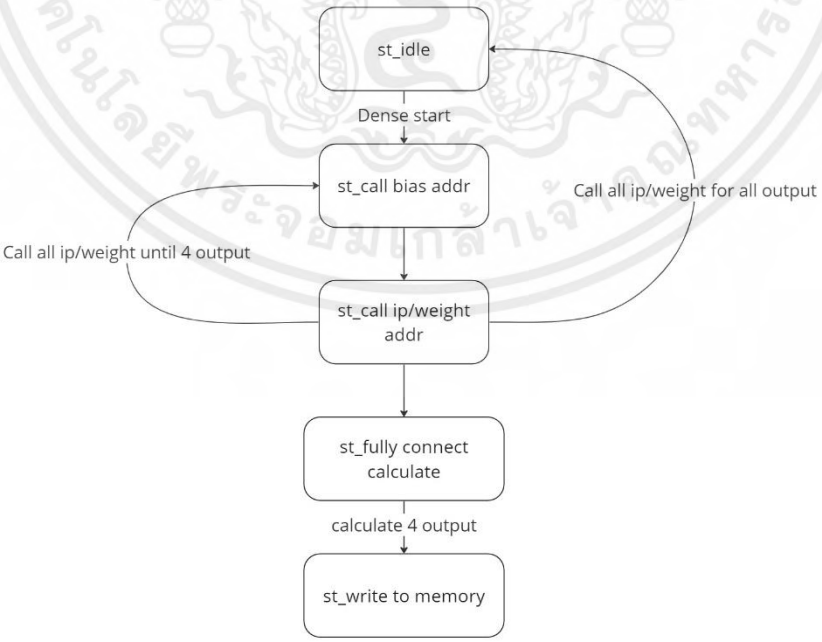
3.5.3 การออกแบบวงจร Dense_Cell

วงจร Dense_Cell เป็นวงจรที่จะนำ input ทั้งหมดมาคำนวณเป็น linear regression ออกเป็น output ครั้งละ 4 ค่าแล้วเขียนลงใน mem i/o เมื่อเริ่มการทำงานจะดึงค่า bias ของ output 4 ค่าจาก mem coef ออกมาเก็บไว้ใน register แล้วจึงอ่านค่า input และ weight อย่างละ 4 ตัวจาก mem i/o และ mem coef ตามลำดับ หลังจากนั้นจึงนำค่ามาใช้ในการคำนวณดังต่อไปนี้



รูปที่ 3.6.3 คำนวณ Fully Connect ใน 1 รอบการทำงานของ Dense_Cell

จากรูปดังกล่าวจะเห็นว่าหลังจากโหลด input และ weight ออกมาจากหน่วยความจำแล้ว input และ weight แต่ละตัวจะถูกจับคูณกันแล้วเก็บไว้ใน register d0-3 แล้วจึงนำ d0-3 มาบวกกันแล้วเก็บไว้ใน register sum0,1 และนำ sum0 และ sum1 มาบวกกันเก็บไว้ใน register sum_all สุดท้ายจึงนำ sum_all มาบวกกับ sum_all ค่าเก่า หรือbias ในกรณีที่เป็นการทำ Fully Connect รอบแรกของการคำนวณ output ตัวนั้นๆ

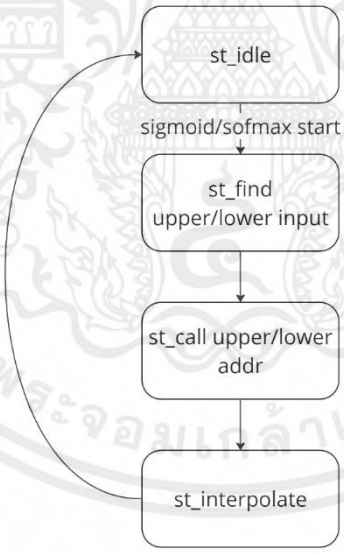


รูปที่ 3.5.4 Stage การทำงานของวงจร Dense_Cell

3.5.4 การออกแบบวงจรที่เป็น activation function

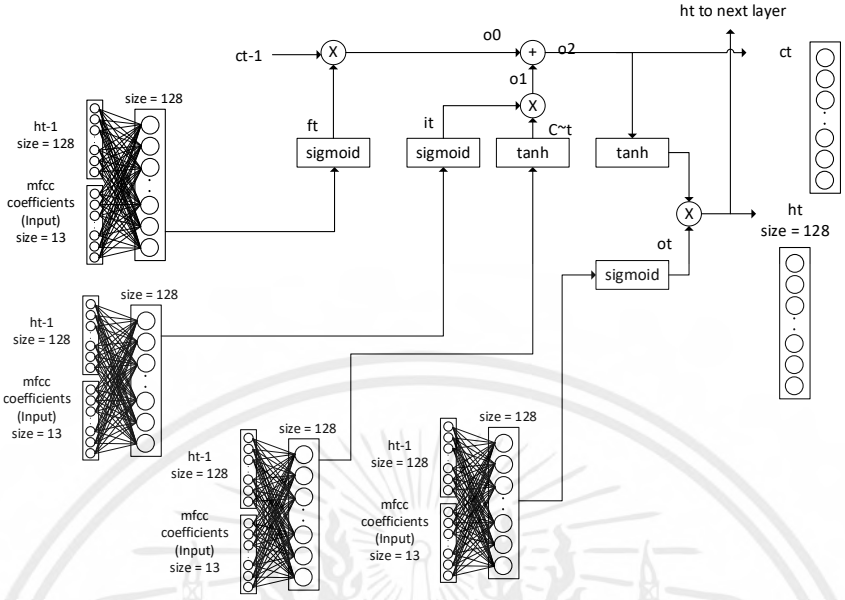
วงจรที่เป็น activation function ใน model นี้สามารถแบ่งออกเป็น 2 ประเภทคือ วงจรที่ไม่จำเป็นต้องอ่านค่าคงที่ภายในหน่วยความจำ และวงจรที่จำเป็นต้องอ่านค่าคงที่ภายในหน่วยความจำ

1. วงจรที่ไม่จำเป็นต้องอ่านค่าคงที่ภายในหน่วยความจำ เป็นวงจรที่มีการคำนวณภายในฟังก์ชันไม่ซับซ้อนและสามารถคำนวณค่าออกมาได้ง่าย สำหรับ model นี้ จะมีเพียงวงจรเดียวคือวงจร relu ซึ่งจะมีฟังก์ชั่นหลักคือการปรับค่าของ input ที่น้อยกว่า 0 ให้กลายเป็น 0 แล้วคงค่า input ที่มากกว่า 0 ไว้ดังเดิม
2. วงจรที่จำเป็นต้องอ่านค่าคงที่ภายในหน่วยความจำ เป็นวงจรที่มีการคำนวณภายในฟังก์ชันซับซ้อนจึงอาจต้องใช้ทรัพยากรจำนวณมากถ้าจะให้มีการคำนวณนั้นเกิดขึ้น จึงเก็บเป็นค่าคงที่ของการคำนวณบางค่าไว้ภายในหน่วยความจำแล้วจึงนำ input ไปคำนวณโดยอ้างอิงจากค่าคงที่ภายในหน่วยความจำที่เป็นผลลัพธ์จากการป้อนค่าที่ใกล้เคียงกับ input 2 ค่า มา interpolate เพื่อประมาณค่า input เมื่อถูกป้อนเข้ามาภายในฟังก์ชันดังกล่าว ซึ่งภายใน model นี้จะมีวงจรจำเป็นต้องอ่านค่าคงที่ภายในหน่วยความจำคือ วงจร sigmoid , softmax,tanh



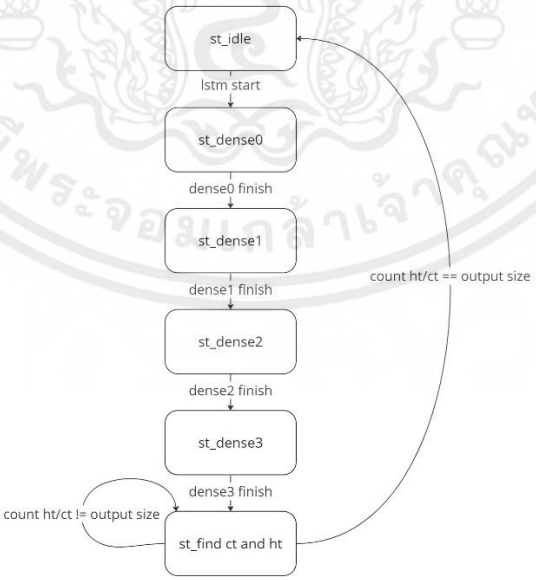
รูปที่ 3.5.5 Stage การทำงานของวงจรที่จำเป็นต้องอ่านค่าคงที่ภายในหน่วยความจำ

3.5.5 การออกแบบวงจร LSTM_Cell



รูปที่ 3.5.6 การทำงานพื้นฐานของ lstm

วงจร LSTM_Cell เป็นการสั่งใช้งานวงจร Dense_Cell เป็นจำนวน 4 ครั้ง หลังจาก output ของ Dense_Cell ทั้งหมดถูกเขียนลงในหน่วยความจำแล้ว ในขั้นตอนต่อไปจะเป็นการคำนวณหา ct และ ht และเขียนลงในหน่วยความจำ ตามรูปการณ์ทำงานพื้นฐานของ LSTM ซึ่งจะมีอ่านข้อมูลจากหน่วยความจำและการสั่งใช้งานวงจร activation function ด้วย



รูปที่ 3.5.7 Stage การทำงานของวงจร LSTM_Cell

บทที่ 4

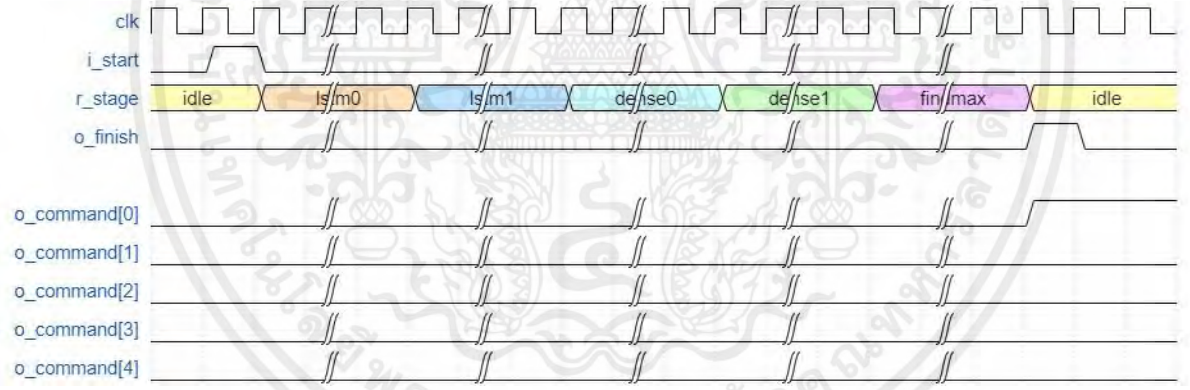
ผลการทดลอง

4.1 ผลจากการ simulation เพื่อตรวจสอบความถูกต้องของการทำงาน

ทำการ simulation โดยมีคำสั่งนึงเข้าไปทุกๆเฟรมเข้าไปในระบบบนชิปที่มีฮาร์ดแวร์เร่งประมวลผล LSTM โดยใช้ model การประผลคำสั่งโดยให้

- command[0] เป็นคำสั่ง “ไป”
- command[1] เป็นคำสั่ง “หยุด”
- command[2] เป็นคำสั่ง “ซ้าย”
- command[3] เป็นคำสั่ง “ขวา”
- command[4] เป็นคำสั่ง “unknow”

เมื่อป้อนคำสั่ง “ไป” เข้าไปใน model ทุกเฟรม



รูป 4.1.1 ผลจากการ simulation frame สุดท้ายเมื่อป้อนคำสั่ง “ไป”

4.2 คำนวณเวลาที่ใช้ในการประมวลผล

ตารางที่ 8 จำนวน Clk ในการทำงานของ mfcc ต่อ 1 เฟรม (Clk 12MHz)

function	จำนวน Clk
Window function (Hamming)	4111
FFT^2	224610
Mel Filter Bank	5727
Power to dB	286
DCT type II	25259
รวม	259993

ตารางที่ 9 จำนวน Clk ในการทำงานของวงจร model ต่อ 1 เฟรม (Clk 12MHz)

Layer	จำนวน Clk
LSTM0	18584
LSTM1	12376
Dense0	1046
Dense1	136
รวม	32142

จากตารางที่ 8 และ 9 จะพบว่าจำนวน Clk ที่ใช้ในการทำงานของ mfcc นั้นมีค่ามากกว่าการทำงานของวงจร model มาก และจากความถี่ Clk แปลงเป็นเวลาที่ประมวลผลได้ดังนี้

$$\text{time} = (259993)\text{x}\frac{1}{12MHz} = 0.02167 \ sec/frame$$

โดยจาก sampling rate 44.1kHz เราใช้ 2048 sample/frame จะได้22 frame จะทำงานได้ที่ความถี่ดังนี้

$$\text{Clk} = (259993)\text{x}22 = 5.719846MHz$$

บทที่ 5
สรุปผลการทดลอง

จากการทดสอบพบว่าการออกแบบระบบบนชิปและฮาร์ดแวร์เร่งประมวลผล LSTM แบบปรับแต่งได้สามารถนำมาใช้ในการประมวลผลเสียงแยะแยะคำตอบได้อย่างถูกต้องผ่านการ simulation และสามารถทำงานได้อย่างรวดเร็วโดยใช้เวลาประมวลผล 0.02167 วินาที ต่อ 1 เฟรม และระบบสามารถทำงานได้ด้วยที่ความถี่ Clk เพียง 6 MHz โดยสามารถออกแบบ ฮาร์ดแวร์เร่งประมวลผล LSTM ให้ปรับแต่งขนาดได้ตามการใช้งาน ทั้งนี้ยังสามารถเปลี่ยนรูปแบบการกรองสัญญาณจากแบบ mfcc ให้เป็นชนิดอื่นได้ผ่านการโปรแกรมเพื่อให้เหมาะสมกับการใช้งานด้านต่างๆได้ดี

บรรณานุกรม

[1] Kasidis Satangmongkol. (2022). เข้าใจการทำงานพื้นฐานของ Neurons ใน Neural Networks, สืบค้นเมื่อ 5 ตุลาคม 2566.

จาก. https://datarockie.com/blog/how-neurons-work/

[2] baeldung. (2023). The Concepts of Dense and Sparse in the Context of Neural Networks, สืบค้นเมื่อ 6 ตุลาคม 2566.

จาก. https://www.baeldung.com/cs/neural-networks-dense-sparse

[3] Pranj52 Srivastava. (2023). Essentials of Deep Learning : Introduction to Long Short Term Memory, สืบค้นเมื่อ 6 ตุลาคม 2566.

จาก. https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/

[4] Anas Al-Masri. (2023). How Does Backpropagation in a Neural Network Work?,

สืบค้นเมื่อ 7 ตุลาคม 2566.

จาก. https://builtin.com/machine-learning/backpropagation-neural-network

[5] Emmanuel Deruty. (2022). Intuitive understanding of MFCCs, สืบค้นเมื่อ 11 ตุลาคม 2566.

จาก. https://medium.com/@derutycsl/intuitive-understanding-of-mfccs-836d36a1f779

[6] Rémy Pujol. (2018). HUMAN AUDITORY RANGE, สืบค้นเมื่อ 14 ตุลาคม 2566.

จาก. https://www.cochlea.org/en/hear/human-auditory-range

[7] Python's BDFL. (2009). HUMAN AUDITORY RANGE, สืบค้นเมื่อ 14 ตุลาคม 2566.

จาก. https://www.cochlea.org/en/hear/human-auditory-range

[8] Steve Radecky. (2023). What is an FPGA?, สืบค้นเมื่อ 14 ตุลาคม 2566.

จาก. https://python-history.blogspot.com/2009/01/personal-history-part-1-cwi.html

[9] Thanadol Singkhornart. (2021). การเรียนรู้เชิงลึกสำหรับการตรวจจับและรู้จำคำบรรยายในวีดิทัศน์, สืบค้นเมื่อ 8 ตุลาคม 2566.

จาก. http://olarik.it.msu.ac.th/wp-content/uploads/2021/11/Thesis_Thanadol.pdf

[10] Kinza Yasar. (2023). neural network, สืบค้นเมื่อ 8 ตุลาคม 2566.

จาก. https://www.techtarget.com/searchenterpriseai/definition/neural-network

[11] Sinauer Associates. (2001). The Audible Spectrum, สืบค้นเมื่อ 9 ตุลาคม 2566.

จาก. https://www.ncbi.nlm.nih.gov/books/NBK10924/