VLSI Implementation of LSTM

A thesis report submitted for BTP phase I

by

Lakshita

(Roll No. 190108030)

Under the guidance of

Prof. Shaik Rafi Ahamed

DEPARTMENT OF ELECTRONICS & ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

November 2021

Abstract

The objective of this project is to implement Long Short Term Memory on a VLSI chip.

Most of the previous works on VLSI architecture for LSTMs focus on implementing the computational part with low complexity and better efficiency and reducing the memory overhead due to which excessive amount of resources are used which results in low throughput . But Neural Networks deal with a huge amount of data and are large in size, hence high throughput is one of the major requirement for their efficient VLSI implementation. This report talks about the efficient techniques to achieve high throughput with low area and power requirements.

Contents

Nomenclature

LSTM	Long Short Term Memory
RNN	Recurrent Neural Network
MVM	Matrix Vector Multiplication
C-MVM	Circulant Matrix Vector Multiplication
DA	Distributed Arithmetic
OBC	Offset Binary Coding
IPC	Inner Product Calculation
MAC	Multiply and Accumulate
PPG	Partial Product Generator
PPS	Partial Product Selector

Chapter 1

Introduction

Long Short Term Memory (LSTM) is a Recurrent Neural Network (RNN) known for its ability to capture long-term dependencies in sequential data. They are the workhorse of the deep learning community for most sequence modeling tasks like speech recognition and synthesis, text generation and can even be used in caption generation for videos.

The main difference between a simple RNN and LSTM lies in the repeating modules. In RNN, these modules comprise a simple computation node, but in LSTM, they contain a computational block that controls the information flow. Hence, their computational complexity is very high due to added MVM. Because of the high number of parameters, the memory demands have also increased widely.

Hardware implementation of neural networks has become high trending nowadays because it works on parallelism and is much faster than software simulations.Even the fastest sequential processor cannot provide real-time response and learning for networks with large numbers of neurons, But Parallel processing with multiple simple processing elements can provide incredible speedups.

Parallelism is achieved by pipelining which comes with the drawback of more power consumption and chip area, But we can optimize it by using different pipelining techniques such as wave pipelining which achieves the goal without inserting registers, the main source of power consumption.

Chapter 2

Literature Survey

LSTMs have increased computational complexity than RNN because of additional matrix-vector multiplication (MVM) and memory requirements for the storage of network parameters. [1] discusses a generalized approach to accelerate C-MVM. The design uses circulant matrices because this can be constructed from a single primitive vector and hence the number of parameters are far less than that in the usual matrix. As a result, it reduces the memory requirements. DA algorithm is used to realize IPC efficiently without using multipliers which are optimized using OBC and then by separating the PPG and PPS units. Because of the use of the circulant matrices, the PPS unit is further optimized by sharing their min-terms across the Boolean expressions. The design is pipelined using Fine-Grained pipelining to achieve one adder delay.

[3] discusses the implementation of MVM and non-linear activation functions using MAC units. [4] describes different learning strategies with their pros and cons. Learning can be done using a simulated network in software which is known as *off-chip learning*. Or it can be done using both the network and external software computation where software performs the learning algorithm but computation is performed by hardware network and it's known as *Chip in the loop*. The third strategy is to use only hardware to learn which is called *On-chip learning*. Off-chip learning performs the training calculations faster with more accuracy but it doesn't take care of hardware manufacturing variations while 'chip in the loop' does take care of that with calculation precision independent of hardware potential. 'On chip learning' is slower and less accurate than the above two methods but it can be updated over time.

[5] discusses the piecewise approximation of hyperbolic tangent and sigmoid functions. And it shows that time taken for the training of the network can be reduced if we use the derivative of piecewise functions instead of using them directly.

Chapter 3

Problem Statement

Memory requirement and computational complexity both are overhead in LSTMs, which brings the need for their efficient VLSI implementation.

Most prior work in this area focuses on decreasing the complexity of MVM and non-linear activation functions. For example, in [1] an OBC-DA based architecture have been presented to perform C-MVM., in [5] a piecewise approximation of activation functions has been done. But these approaches fail to achieve a high throughput VLSI architecture, which is a critical parameter to measure the performance of any VLSI design.

The LSTM networks are so large that sometimes, they don't satisfy resource constraints. Hence, low throughput becomes a critical issue for such networks. So, our goal is to get a "high throughput" efficient VLSI architecture for LSTMs.

Chapter 4

Long Short Term Memory

LSTM uses gates to control the flow of information. *Forget* gate gets rid of irrelevant information, *Output* gate controls what information is sent to the next time step, Input gate decides the relevant information to add from the current state and *Cell* carry relevant information from one stage to another.

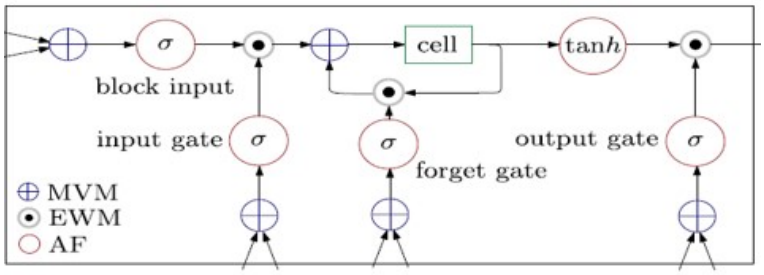

Figure 4.1: LSTM Cell

The basic LSTM cell shown in above figure is characterized by following equations:

$$i^t = \sigma(W_i x^t + R_i y^{t-1} + b_i) \tag{4.1}$$

$$f^t = \sigma(W_f x^t + R_f y^{t-1} + b_f) \tag{4.2}$$

$$o^t = \sigma(W_o x^t + R_o y^{t-1} + b_o) \tag{4.3}$$

$$z^t = \sigma(W_z x^t + R_z y^{t-1} + b_z) \tag{4.4}$$

$$c^t = z^t . i^t + c^{t-1} . f^t \tag{4.5}$$

$$y^t = \tanh(c^t) . o^t \tag{4.6}$$

Chapter 5

Throughput

Throughput is the rate at which inputs or outputs are proceessed in any digital system. It is one of the key parameters to measure the performance of any VLSI Design. Therefore it is essential to have high throughput for Efficient designs.

5.1 Techniques to achieve High throughput

One of the famous method to increase throughput is pipelining which increases the number of task that can be done concurrently. But ”conventional or fine grained pipelining” (used in [1]) inserts registers or latches to achieve high throughput which brings the drawback of high power and area requirement.

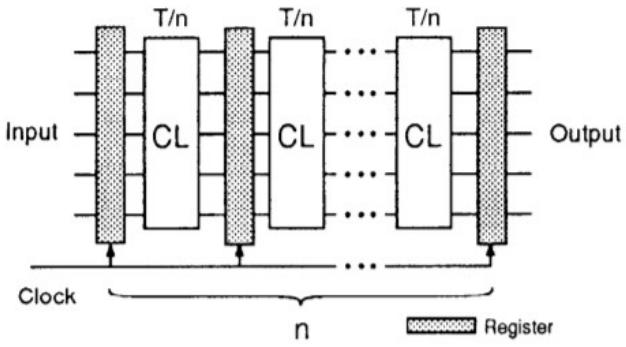

Figure 5.1: Conventional Pipelining

This problem can be solved by ”Wave pipelining” which works on the difference between propogation and contamination delays of the system instead of only the longest path delay. it

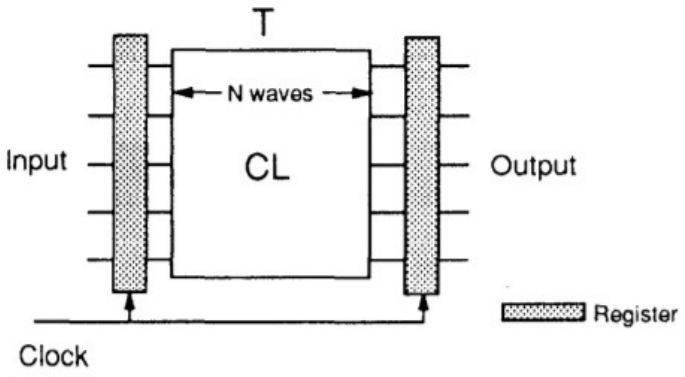

Figure 5.2: Wave pipelining

achieves the goal of pipelining without inserting clocked elements and hence it decreases the power and area requiremnt by a huge amount as compare to conventioanal method. But wave pipelining proves to be useful only when the difference between the longest and shortest path delays is almost 3 or 4 times less than the clocking period of system.

Hence the optimum combination of both the above techniques is the best choice for achieving better performance with high-throughput which is known as ”optimal pipelining”. Few registers are put between a sets of combinational logic which keeps the delay difference such that wave pipelining can work efficiently. In this way, both techniques optimize each other and results in a high-throughput design minimizing the downsides of conventional methods.

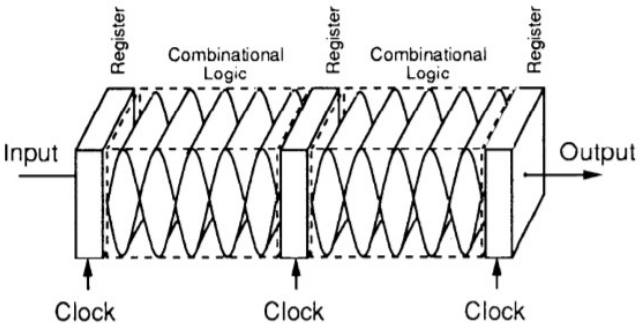

Figure 5.3: Optimal Pipelining

Chapter 6

Conclusion and Future Work

This project aims to implement a ”Long Short Term Memory” network on a VLSI chip. This report reviews the prior work done in this field and comes at the conclusion that there is a strong need to work for getting High-throughput VLSI architectures for LSTMs. Then it discusses some pipelining techniques which can be used to increase throughput with high design efforts. Since we analyzed that optimal pipelining is the best approach to achieve high throughput, We will use this technique to pipeline our VLSI architecture for LSTM.

Further modules of this project include Optimal pipelining based architecture for MVM computation, implementation of activation functions such as tangent or logistic sigmoid to regulate the information flow, on-chip training using the algorithms like Back Propagation Through Time(BPPT), Simultaneous Perturbation Stochastic Approximation (SPSA) [2] with a special focus on getting High-throughput.

Bibliography

[1] Krishna Praveen Yalamarthy; Saurabh Dhall; Mohd. Tasleem Khan; Rafi Ahamed Shaik, "Low-Complexity Distributed-Arithmetic-based Pipelined Architecture for an LSTM Network", *IEEE Transactions on Very Large Scale (VLSI) Systems, Vol. 28, No. 2, pp. 329-338*, Feb. 2020.

[2] "FPGA Implementation of LSTM Neural Network, [Online]. Available: https://repositorio-aberto.up.pt/bitstream/10216/90359/2/138867.pdf

[3] A. X. M. Chang, B. Martini, and E. Culurciello, "Recurrent neural networks hardware implementation on FPGA," 2015, arXiv:1511.05552. [Online]. Available: https://arxiv.org/abs/1511.05552

[4] Mats Forssell, "Hardware Implementation of Artifical Neural Networks",

[5] I. Kouretas and V. Paliouras, "Hardware Aspects of Long Short Term Memory",*IEEE Transactions on Electronics, Circuits and Systems* 2018

[6] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016, http://www.deeplearningbook.org.

[7] "Wave-Pipelining: A Tutorial and Research Survey" *Wayne P. Burleson, Member, IEEE, Maciej Ciesielski, Senior Member, IEEE, Fabian Klass, Associate Member, IEEE, and Wentai Liu, Senior Member, IEEE*

[8] "Comparative Studies of Pipelined Circuits", *Fabian Klass and Michael J. Flynn, Technichal Report No. CSL-TR-93-579, Standford University*