VLSI Implementation of LSTM

A thesis report submitted for BTP phase II

by

Lakshita

(Roll No. 190108030)

Under the guidance of

Prof. Shaik Rafi Ahamed

DEPARTMENT OF ELECTRONICS & ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

April 2022

Abstract

The most computationally expensive part of an LSTM is Matrix Vector Multiplication. Hence, increasing the efficiency of MVM architecture can lead towards the high throughput implementation of whole LSTM Network. Pipelining techniques like wave pipelining can increase the throughput of the circuit efficiently without inserting intermediate registers or latches which result in less power and area consumption as compared to normally pipelined circuits.

Contents

Nomenclature

LSTM	Long-Short Term Memory
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

Background

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

MVM is just an array of inner product computation (IPC) of two vectors. A famous multiplier-less method to efficiently realize IPC is Distributed arithmetic algorithm. It uses a lookup table with a shift-accumulator. Although DA provides sufficient relaxation in chip area and power consumption, it has the drawback of large LUT size and exponentially growing access time with an increase in IPC order. Hence, its complexity is reduced using offset binary coding.

Hardware implementation of neural networks works on parallelism and is much faster than software simulations.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 intermediate registers, the main source of power consumption. It works on the difference between propagation and contamination delays of the system instead of only the longest path delay.

Chapter 2

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. High throughput can be achieved by inserting intermediate registers and pipelining the circuit conventionally but that results in increased power and area demand. Hence, the focus of this report is " Wave Pipelined Architecture for MVM computation in LSTM Networks".

Chapter 3

Wave Pipelining

In conventionally clocked digital circuits, the minimal clock period depends on the maximum path delay. But, individual logic gates remain idle for majority of each clock cycle. To understand it better, let's consider the modern CMOS circuits where the gate switching time is generally near 1 or 2 ns while the clock period is approx 10ns. It shows that, for approx 90-95 % of the clock time, we are not using the individual logic gates. We can improve the utilization of resources by minimizing this idle time by using a technique called wave pipelining. The following figure represents the dataflow through a non-pipelined combinational logic

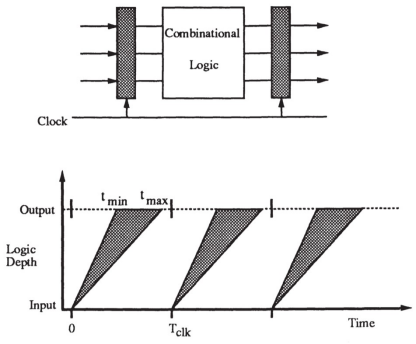

Figure 3.1: Non-Pipelined Combinational Logic

In a standard system, clocking of output register is done once the latest data has arrived with some allowance for setup time of register, which has been represented by un-shaded region, because the logic remains stable in this region. Hence, minimum clock period which is needed, is limited by the propagation delay of combinational logic, register's setup and hold time and the clock skew present between input and output registers.

To decrease the clock period and increase the utilization of available resources, Conven-

tionally, we insert some intermediate registers and pipeline the logic. The following figure depicts this method.

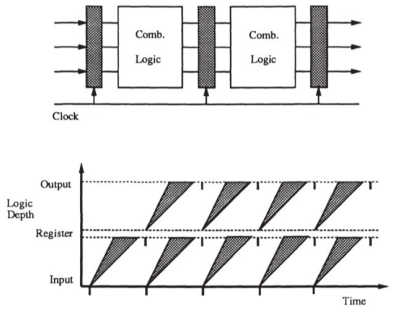

Figure 3.2: Non-Pipelined Combinational Logic

Here, we have divided the the complete logic in two different parts by inserting an intermediate register. It allows the circuit to take two clock cycles for computation and provide result after each clock cycle. Since, the max delay of combinational logic has been decreased here, the min clock requirement also gets reduced. But the propagation delay, hold and setup time and clock skew present between the registers act as a counterpart. In addition, more registers implies increasing demand on clock distribution network which results in increased area-requirement and power-consumption of the design.

In figure 3.1, we can see that logic remains stabilized for a long section of clock-period, which indicates that the combinational circuit is not being operated at its maximum speed. Hence, there is a sufficient scope to reduce the clock period further as long as the data uncertainty regions doesn't overlap. The following figure depicts this approach.

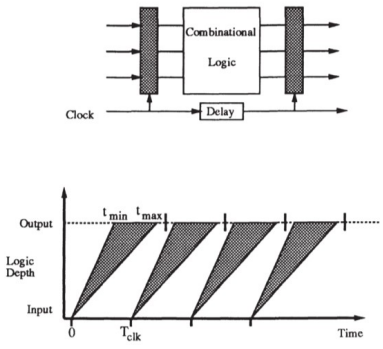

Figure 3.3: Wave-Pipelined Combinational Logic

Now, the restriction on time period of the clock is set by propagation delay difference present between maximum and minimum path along with setup-hold time and clock skews.

3.1 Clocking Constraints

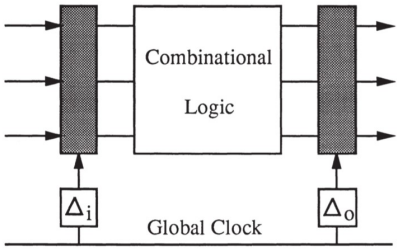

Figure 3.4: Single Stage Block Diagram

Parameters

Circuit Timing Parameters

t_{max}/t_{min} = max/min logic delay present between i/p and o/p register

$t_{max}(i)/t_{min}(i)$ = max/min logic delay present between i/p register and i^th internal node

$t_{stable}(i)$ = minimum time required for stabilizing the i/ps at the gates of i^th internal node.

Clock Timing Parameters

T_{clk} = clock period

Δ_i = input clock delay

Δ_o = output clock delay

Δt_e^n = earliest clock-skew at register of i/p stage

Δt_l^n = latest clock skew at register of i/p stage

Δt_e^o = earliest clock skew at register of o/p stage

Δt_l^o = latest clock skew at register of o/p stage.

Register Timing Parameters

t_d = propagation delay of register

t_{setup} = setup-time of register

t_{hold} = hold-time of register.

3.1.1 Register Constraints

The time period of the clock must be chosen such that the data at output register is latched only when the latest data has reached and the earliest data of next clock cycle is yet to arrive.

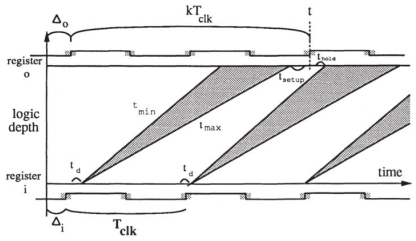

Figure 3.5: Register Constraint

This constraint is given by the following equation:

$$T_{clk} \geq (t_{max} - t_{min}) + t_{setup} + t_{hold} + \Delta t_e^n + \Delta t_e^o + \Delta t_l^n + \Delta t_l^o \tag{3.1}$$

3.1.2 Internal Node Constraints

At any i^th node, the succeeding earliest feasible data should reach only when the propagation of the latest possible data through that node has been finished. This condition can be satisfied by ensuring that each logic gate input data is stable for a sufficient amount of time so that the logic has enough time to respond before the arrival of the next data. The following equation describes the above constraint.

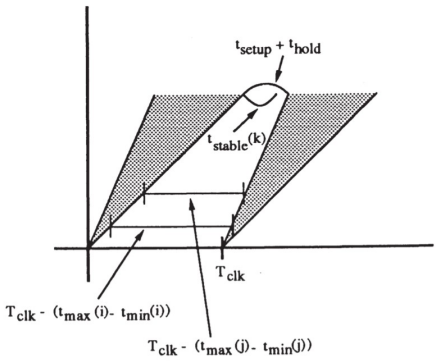

Figure 3.6: Interanal Node Constraint

$$T_{clk} \geq \max_{(i,j)\in C}(t_{max}) - \min_{(i,j)\in C}(t_{min}) + t_{setup} + t_{hold} + \Delta t_e^n + \Delta t_e^o + \Delta t_l^n + \Delta t_l^o \tag{3.2}$$

3.2 Tuning Techniques

We can tackle the problem of minimizing path delay differences in two ways. One is insertion of delay elements and other is adjustment of gate drives. These two methods have been named as Rough tuning and Fine tuning. Rough tuning has the drawback of increased area requirement while fine tuning increases power consumption.

3.2.1 Rough Tuning

In this tuning method, the delay elements(buffers or inverter chains) are inserted to extend the total path delay. To decrease the area and power consumption of the circuit, it becomes prominent to reduce the number of buffers added. To ensure that the circuit can be balanced by just adjusting the gate drives, Rough tuning is generally used to insert minimal buffer elements.

3.2.2 Fine Tuning

This method of tuning is all about adjusting the output driving transistors' size so that propagation delay through a gate can be manipulated. When using fine-tuning, we need to keep a note that gates having different rise and fall delays can influence all paths which are using them. To equalize the rise and fall delays, the P-N ratio of the transistor should be adjusted.

These two methods actually complements each other.In a general case, we combine these two tuning methods to balance the circuit,

3.3 Design Challenges

Wave Pipelining is fully dependent on the sensitivity of design to path delay differences. But it is always possible to minimize the difference using tuning techniques discussed above.

The clock frequency can also be limited by disordered clock skew. Therefore, to get efficient performance, the clock skew should be minimized. Since wave pipelined circuits operate on high frequencies, the registers and latches should be of small sampling time. For which, intrinsic delay of latches/registers needs to be considered specifically along with the setup and hold time. Logic gates having low fall and rise time should be prioritized to cover the pipeline. Along with that, they must have small fanout so that capacitance can be reduced.

Chapter 4

Conclusion and Future Work

This work discusses the basic concepts of wave pipelining along with clocking constraints, detailed timing analysis, tuning techniques and design challenges that need to be considered. The design process of wave pipelined circuits require standards that are normally unknown until we implement the design. To balance the paths, we need to know the design parameters such as path delays etc. which can be achieved by using some available scripts and tools.

Future work directs towards the practical consideration like architectural properties, delay variations of paths, selecting the logic families to maximize the performance of wave pipelining. The focus will be on modifying the OBC-DA based MVM computation architecture [1], according to the need of wave pipelined circuits. Then, implementation of both rough and fine tuning techniques needs to be done because they both optimize each other which can provide an efficient design with minimized drawbacks like power and area consumption.

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] Keshab K. Parhi, 2013. "VLSI Digital Signal Processing Systems Design and Implementation" John Wiley Edition.

[3] C. Thomas Gray, W. Liu and R. Cavin, "Wave Pipelining: Theory and CMOS implementation," *Kluwer Academic Publishers, 1993*.

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

[5] "Valid Clocking in Wave Pipelined Circuits" *William K.C. Lam, Robert K Brayton, Albert0 L. Sangiovanni-VincenteIli, University of California Berkeley, CA 94720*

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

[7] Some Experiments About Wave Pipelining on FPGAs *Eduardo I. Boemo, Sergio L´opez-Buedo, and Juan M. Meneses*

[8] "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*

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

[10] Vireen, V., G. Seetharaman and B. Venkatramanani, 2008. "Synthesis Techniques for Implementation of Wave -Pipelined circuits in ASICs"International Conference on Electronic Design, Penang, Malaysia.