FPGA Implementation of LSTM Based on Automatic Speech Recognition

Chen-Lu Li, Yu-Jie Huang, Yu-Jie Cai, Jun Han, Xiao-Yang Zeng

[1] Department of Microelectronics, Fudan University, Shanghai 200433, China
[2] State Key Laboratory of ASIC and System, Fudan University, Shanghai 200433, China
* Email: clli16@fudan.edu.cn

Abstract

Automatic speech recognition(ASR) remains very popular research area in recent years.Long Short-Term Memory(LSTM) is always used in speech recognition and demonstrated state-of-art accuracy. The high performance LSTM models is becoming increasing demanding in terms of computational and memory load. FPGA-based accelerators have attracted attention of researchers because of its high energy-efficiency and great flexibility.The main work of this paper is an efficient FPGA implementation of LSTM to accelerate the ASR algorithm.The proposed circuit is implemented at Zedboard 100 MHz.The amount of parameters are reduced more than 7 times because of compression method.

1. Introduction

The main aim of Automatic Speech Recognition (ASR) system is the transcription of human speech into words.It is a very challenging task because human speech signals are highly variable due to speaker attributes and environmental noises.Recurrent neural network(RNN) is always applied in ASR system because of its good performance on processing timing information.

Traditional RNN could not maintain long-term memory so the prediction accuracy is not very satisfactory because of the vanishing of gradient problem. LSTM is the most popular varient of RNNs in nowadays, it was first designed in [1] as a memory cell to decide what to remember, what to forget and what to output. A typical basic LSTM architecture is showed in Figure 1.The operation in LSTM is given by the following set of equations.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \quad (2)$$

$$\tilde{c}_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (3)$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tilde{c}_t \quad (4)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \quad (5)$$

$$h_t = o_t \otimes \tanh(c_t) \quad (6)$$

Here the big dot operator means element-wise multiplication,the W_* terms denote weight matrices. The b terms denote bias vectors, σ means sigmoid function.

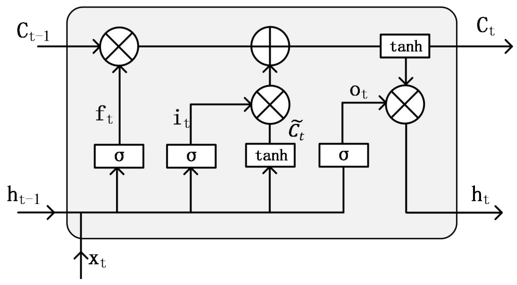

Figure 1. LSTM

2. Related Work

Since a large-scale LSTM contains vast parameters and complex computational logic,there are many problems such as computational efficiency,storage consumption and power consumption.The acceleration of LSTM is a hot issue in today's research.Many efforts have been done to implement LSTM algorithm using hardware,such as GPU,ASIC and FPGA.FPGA-based acceleration has achieved very high performance for LSTM because of its high energy-efficiency and great flexibility.

Guan et al.[2] proposed an FPGA-based LSTM accelerator optimized for speech recognition on a Xilinx VC707 FPGA platform.Chang.[3] completed 2 hidden layers LSTM design which has 128 hidden units every layers implement in Zynq 7020.Song Han[4] proposed an excellent pruning method to compress networks.

This paper makes following contributions:

1) In software,we implement an ASR system based on Connectionist Temporal Classification(CTC) and implement compression methods to save resources;

2) We optimize the LSTM accelerator to meet the need of computation speed performance and resource utilization;

3) At the architecture level,we set up the whole platform(Figure 2),which can fetch input data and send output data from ARM DDR memory.

3. Proposed Architecture

3.1 System Architecture

The overall system architecture are showed in Figure 2. In the designed system, input vectors,output vectors and weight matrices are stored in the DDR3 memory.The modules on-chip are connected with AXI bus and AXI-Lite bus that the AXI bus will send the vectors and parameters to our LSMT PE and the AXI-Lite bus will generate control signals to control the process of data reading and sending.The LSTM PE is packaged as a hardware IP.And we use ARM to initialize the LSTM PE,measure calculating time and control the data communication.

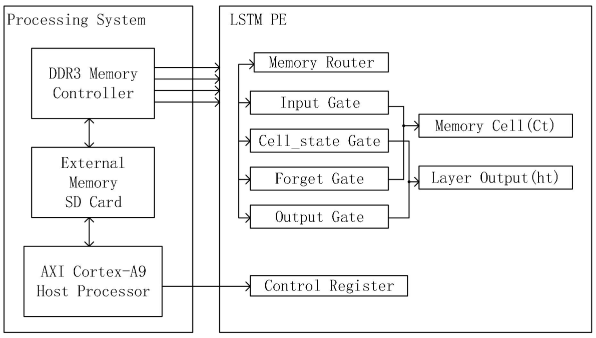

Figure 2. Designed System Diagram

3.2 Gate Module

A complete LSTM has 4 gate modules which are input gate,forget gate,output gate and cell-state gate.The gate modules are responsible for producing internal signal vectors for i_t, f_t, o_t and $\tilde{c}_t$.Each module has the same calculate process,but the inputs are different.The algorithm of the four gates is introduced in 2.1

A. Matrix-Vector Multiplier

As we can see,in the LSTM algorithm,input vector x_t and the hidden layer output of previous time step $h_{t\text{-}1}$ will be multiplied by weight matrices W_* (Width * Height).So we need a HDL block to implement matrix-vector multiplication.

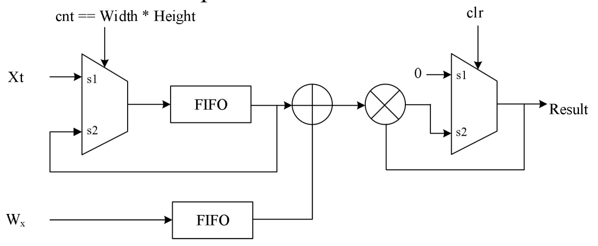

Figure 3. Matrix-Vector Multiplier

This module has two data streams which are input vector stream and weight matrix stream.The every input vector element will be multiplied and added to the elements at each position in the weight matrix to get the output vector.In our design,the length of x_t is 672, W_* is a (672*600) matrix.So our input vector will be stored in the FIFO and be reused 600 times until finish the whole output vector calculating.The build-in counter will generate clear signal. Figure 3 shows the circuit structure of matrix-vector multiplier.

Every element of input vectors requires one multiplication.In our solution ,we have four MAC units in parallel.So we can process 4 sets of data at the same time.The parallel computing process of data is shown in Figure 4.Because of the independence of four gate modules,the gate modules can be calculated in parallel,too.This method improves the computational efficiency of circuit.

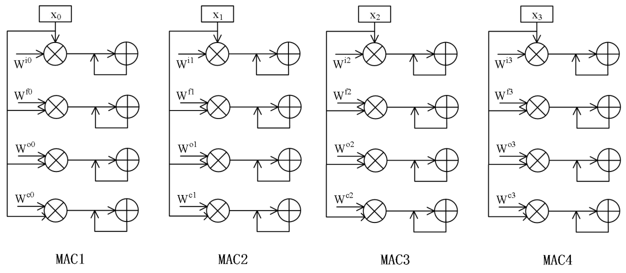

Figure 4. Data Parallel

B. Non-linear activation Function

There are two non-linear activation functions in LSTM,which are sigmoid(x) and tanh(x).These two functions consume a large amount of hardware resources for implementation because of the exponentiation and floating-point divisions. Piecewise linear method has been used which can be implement in FPGA easily.

We store the two non-linear functions approximation in a single-port ROM ,so we can implement this unit easily by a look-up table.

3.3 Element-wise Module

We got four vectors of LSTM by gate modules(i_t , f_t , o_t and $\tilde{c}_t$). And we should use these four vectors to get cell-state vector and output vector of hidden layer by element-wise multiplication and addition.Figure 5 shows the element-wise module architecture.

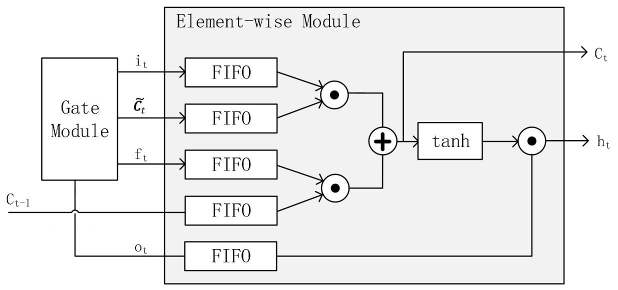

Figure 5. Element-wise Module

Table 1. Recourse Utilization Report

All on chip resource	LUT (53200)	BRAM (140)	DSP (220)	LUT (17400)	FF (53200)
LSTM	6785	24	34	16	3591
AXI_intercon	8820	0	0	730	3529
Processing System	112	0	0	0	0
Periphral	448	0	0	67	232
All(%)	30	17	15.44	5	14

3.4 Compression

As we all known,LSTM algorithm will need large amount of parameters.So how to compress LSTM network without no training accuracy loss is a matter that we concerned.There are two methods to compress our LSTM network,which are UV Decomposition and Fixed-point data conversion.

After UV decomposition,we can compress matrix used in our LSTM PE circuit.As long as we choose the right number of eigenvalues,we can achieve network compression without affecting accuracy. In our solution, our network can be compressed 2.1 times by UV decomposition method.

In traditional neural networks algorithm,the floating-point data are adopted. But corresponding hardware implementation,we need fixed-point operation to reduce computation resources and on-chip data space.

Figure 6 shows the trend of networks' WER(Word Error Rate) under different quantization ways and different number of eigenvalues.We choose 185 eigenvalues and 8-bits fixed-point quantization which result more than 7 times compression in our networks with little accuracy loss.

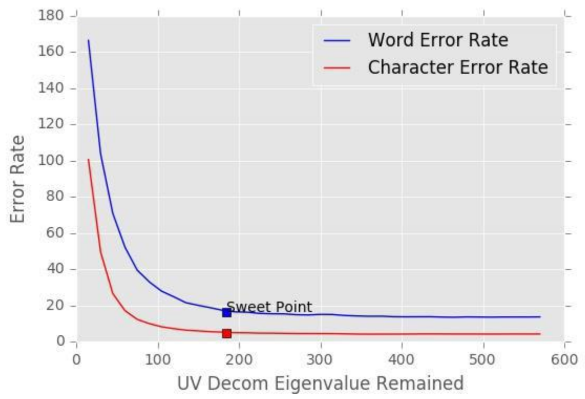

Figure 6. WER of ASR system in different numbers of UV Decomposition eigenvalue and different quantization ways

4. Result and Comparison

Our design is implemented in Zynq-7000 SOC XC7Z020 on 100MHz with hardware and software co-design.Table 1 shows the resource utilization of our architecture.The time to complete a LSTM PE is 0.0099s compared to 0.38s in CPU.The amount of parameters is reduced from the original 59.8MBytes to 8.25MBytes (more than 7 times) because of compression method. EER(Energy Efficient Ratio) of one LSTM layer is 670Mop/s/W.

5. Conclusion

In this paper,our main contributions are:

1) Build ASR system to achieve 0.23 WER and 0.08 CER(Character Error Rate) in LibriSpeech clean data set and 0.34 WER and 0.13 CER in CHiME noisy data set;

2) Map the LSTM algorithm in Zedboard ;

3)Proposed an efficient design of matrix-vector multiplier and non-linear activation computation of LSTM and proposed compression method to save more than 7 times design resources.

Acknowledgments

The authors would like to thank students and teachers in State Key Laboratory of ASIC & System, Fudan University.This work was supported by the National Natural Science Foundation of China (61574040, 61525401, 61751401).

References

[1] Bengio Y, Simard P, Frasconi P. Learning long-term dependencies with gradient descent is difficult[J]. IEEE Transactions on Neural Networks, p.157(2002).

[2] Guan Y, Yuan Z, Sun G, et al. FPGA-based accelerator for long short-term memory recurrent neural networks[C].Design Automation Conference. IEEE, p.629-634(2017).

[3] Chang Andre Xian Ming, Martini B, Culurciello E. Recurrent Neural Networks Hardware Implementation on FPGA[J]. Computer Science, (2015).

[4] Han S, Kang J, Mao H, et al. ESE: Efficient Speech Recognition Engine with Sparse LSTM on FPGA[J]. (2017).