Long Short-Term Memory Acceleration on RISC-V Rocket CPU

Cristian Hellmer
North Carolina State University
11/23/21

Summary and Introduction

Long Short-Term Memory (LSTM) is an artificial class of Recurrent Neural Network (RNN) architecture. LSTM is used by many companies in the field of deep learning and those companies often utilize dedicated hardware to accelerate those computations. The LSTM unit in this project takes vectors of 16 shorts as vectors for each time step and uses these to compute the output. These vectors include an input vector $x^{(t)}$, the previous cell state vector $c^{(t-1)}$, and the previous output vector $h^{(t-1)}$. These vectors are used to compute the cell state vector $c^{(t)}$ and the output vector $h^{(t)}$ for current time step being calculated. The LSTM unit can accommodate 16 time-step transactions in its memory at a time. The task of this project was to implement a simulation of an SoC with one LSTM unit using TLM in SystemC and use it to process 32 time-steps of a 2-layer LSTM network. The C-program running on the CPU writes memory locations in a DMA unit which signify where to find the inputs and where to write the outputs. The DMA can write memory locations in DRAM or the LSTM unit. The inputs, weight matrices, and bias vectors are stored in DRAM before the simulation begins. At the end of the program, the C-program checks to make sure that the calculation was performed correctly by comparing the output in DRAM to expected outputs.

System Implementation

The final solution that I implemented involves the CPU sending multiple DMA transactions from the DRAM to the LSTM unit and vice versa to send data to and from the LSTM unit. During the course of the C-program, the CPU requests the following actions to be made by the DMA and LSTM unit.

- Have the DMA send the layer-1 coefficients stored in the DRAM to the LSTM unit.
- Have the DMA send the first 16 time-steps of input data stored in the DRAM to the LSTM unit.
- Initiate the calculation on the LSTM unit using the ctrl port.
- Have the DMA send the C result from the LSTM to the DRAM.
- Have the DMA send the H result from the LSTM to the DRAM.
- Have the DMA send the second 16 time-steps of input data stored in the DRAM to the LSTM unit.
- Have the DMA send the C value stored in the DRAM for the initial time-step to the LSTM unit.
- Have the DMA send the H value stored in the DRAM for the initial time-step to the LSTM unit.
- Initiate the calculation on the LSTM unit using the ctrl port.
- Have the DMA send the C result from the LSTM to the DRAM.
- Have the DMA send the H result from the LSTM to the DRAM.

- Have the DMA send the layer-2 coefficients stored in the DRAM to the LSTM unit.
- Have the DMA send the first 16 time-steps of H which from layer 1 stored in the DRAM to the LSTM unit.
- Initiate the calculation on the LSTM unit using the ctrl port.
- Have the DMA send the C result from the LSTM to the DRAM.
- Have the DMA send the H result from the LSTM to the DRAM.
- Have the DMA send the second 16 time-steps of input data stored in the DRAM to the LSTM unit.
- Have the DMA send the C value stored in the DRAM for the initial time-step to the LSTM unit.
- Have the DMA send the H value stored in the DRAM for the initial time-step to the LSTM unit.
- Initiate the calculation on the LSTM unit using the ctrl port.
- Have the DMA send the C result from the LSTM to the DRAM.
- Have the DMA send the H result from the LSTM to the DRAM.

In between all the DMA transactions and LSTM calculations, the CPU polls the status register of the respective component until the status register becomes zero. The status register is set to be one in the DMA and LSTM unit when the action being performed is begun and zero when the action is complete. After all the transactions have been completed, the C-program running on the CPU checks the values outputted by the LSTM module and calculates a cumulative error which is then displayed.

In the LSTM unit, bytes of memory can be written to and read from the local memory in the LSTM. When the address 0x70010008 is written to, the calculation begins and the status byte at location 0x70010000 is changed to one to indicate to outside components that the calculation is in progress. At the end of the calculate() function, the status byte is set to one to indicate to outside components that the calculation has completed. In the calculate() function, there is a programmed 8481 cycle delay to model the delay of the computation. To accomplish the computation, the individual bytes of data which correspond to a single 16-bit value are combined and placed in a corresponding array. There are multiple arrays which are used to hold data such as input, c, h, and the coefficients. These arrays are then used as arguments to the calculate_layer() function which uses the arrays to perform the LSTM operations with functions such as saturate(), sigmoid_fxp(), and tanh_fxp(). After the calculations are performed for each time-step, the data in the c and h arrays, which hold the output of the calculations, are placed into the local LSTM memory at the locations of c and h. Finally, the status byte is changed to zero and the calculate function returns.

To ensure that the speed of the memory was 1 GHz, the memctl module was changed from a 2 GB DDR DRAM to an 8 GB DDR4 DRAM which is capable of operating at 1 GHz. To ensure that the speed of the DMA was 1 GHz, the delay used for each cycle in read and write transactions was changed to 1 ns. To ensure the speed of the LSTM unit was 1 GHz, the delay for each cycle was changed to 1 ns for reading and writing. To model the delay for the LSTM unit, the number of cycles needed to complete transactions was calculated by using the length of the transaction and the 64-bit bus to determine the fewest number of cycles needed. The number of cycles was then used in

conjunction with the wait() function to model these delays. To model the delay of the calculation, the number of cycles that an LSTM unit implemented in RTL was imported into my design. Using the same delay found in an RTL design of the LSTM unit ensures that the delay used is realistic. This number of cycles was used in conjunction with the wait() function to model the delay of the calculation following setup.

To implement the AXI 64-bit crossbar-style buses, I added two mutexes to each of the buses corresponding to the two port ids which are possible in this design. The mutex for each port id is locked before the transport and unlocked after the transport to only allow one transport to each port on each cycle.

Conclusions

Using the dedicated LSTM unit and a 1 GHz clock as opposed to running the LSTM code on the CPU and using a 100 MHz clock, the simulation time was reduced from 12,084,460 ns to 85,066 ns. This is a 142x improvement using the LSTM unit and increasing the speed of the modules to 1 GHz. This speedup shows the effectiveness of using dedicated hardware to complete frequently used calculations as opposed to running the calculations on a general-purpose CPU. The improvement is even more impressive when taking the print statements in the C-program code into account. The print statements were used to introduce intentional delay in the program's execution to avoid errors using the DMA module. The DMA read and writes to the LSTM module also now model realistic delays as opposed to before where the time to read and write to the LSTM module only took one cycle. In my design, it takes 528 ns to transfer the coefficients to the LSTM module from the DMA. This is realistic because it would take that many cycles to send over 4 KB of data over a 64-bit bus. The subsequent transfer of 512 bytes takes 64 ns which is also the number of cycles necessary to send 512 bytes over a 64-bit bus. To assist in observing delays, messages printed by the code include the amount of time key operations took, such as LSTM calculations and DMA transfers to and from the LSTM unit.