An Ultra-Low-Power Serial Implementation for Sigmoid and Tanh Using CORDIC Algorithm

Yaoxing Chang*†, Petar Jokic*, Stephane Emery* and Luca Benini†
*CSEM SA, Switzerland; †ETH Zurich, Switzerland; Email: yaoxing.chang@csem.ch

***Abstract*—Activation functions (AFs) such as sigmoid and tanh play an important role in neural networks (NNs). Their efficient implementation is critical for always-on edge devices. In this work, we propose a serial-arithmetic architecture for AFs in edge audio applications using the CORDIC algorithm. The design enables to dynamically trade-off throughput/latency and accuracy, and possesses higher area and power efficiency compared to conventional methods such as look-up table (LUT) and piece-wise linear (PWL)-based methods. Considering the throughput difference among the designs, we evaluate average power consumption taking into account active and idle working cycles for same applications. Synthesis results in a 22nm process show that our CORDIC-based design has an area of 545.77 µm^2 and an average power of 0.69 µW for a keyword spotting task, achieving a reduction of 36.92% and 71.72% in average power consumption compared to LUT and PWL-based implementations, respectively.**

***Index Terms*—CORDIC, edgeML, sigmoid, tanh, serial architecture**

I. Introduction

Neural networks (NNs) are becoming ubiquitous among edge devices, where area and power efficiency are the key design targets. While spatial NNs often use simple activation functions (AFs), recurrent NNs require more complex sigmoid and tanh functions in audio applications. However, implementations of such non-linear functions are generally hardware-intensive. Various methods have been investigated for implementing AFs, such as look-up table (LUT)-based methods [1], piece-wise linear (PWL) and non-linear approximations [2], and CORDIC-based approaches [3]. LUT AFs use a set of registers to approximate the target function with finite values (entries). Higher accuracy can be attained with more entries at the cost of a dramatic growth of hardware resources (and power consumption). Instead, one can store only the coefficients of a piece-wise function and use arithmetic units to calculate proximal results. Less storage is needed in PWL AF for comparable accuracy, while additional computing units such as multipliers and adders will add extra costs. CORDIC AFs use only hardware-friendly shift and add operations [4]. Fewer registers and arithmetic units are required, at the cost of lower throughput and higher latency due to their iterative approximation.

Various works have proposed solutions to the major drawbacks of the CORDIC methods, *i.e.* latency and throughput. In [3], a high-speed pipeline implementation is proposed where extra LUTs are applied to reduce the iterations. In [5], a high-throughput pipeline implementation with adjustable precision is presented, and in [6], extra adders and dividers are deployed, where the CORDIC block only computes hyperbolic sine and cosine functions. However, the throughput and latency of the AF blocks are generally not the bottlenecks in audio applications due to their low signal bandwidth. The inference time is dominated by the multiply-accumulate (MAC) operations in the weighted sum that precedes AFs in the computation of neurons' activations. Instead, power consumption is more critical, especially for always-on blocks on edge devices. Yet these works do not cover approaches for low-power design and fully exploit its area and power efficiency.

In this work, we propose an ultra-low-power serial-arithmetic architecture for sigmoid and tanh using the CORDIC algorithm. We compare the performance with implementations of a LUT-based method and a PWL-based approximation and evaluate the arithmetic error under different quantization and iteration settings. We further investigate the inference accuracy on LSTM models in a keyword spotting (KWS) task.

II. Methods and Implementation

To implement CORDIC AF, we derive the sigmoid function as (1), where $\cosh(x)-\sinh(x)$ can be calculated by CORDIC in hyperbolic rotation mode, and $\frac{1}{x}$ can be obtained in linear vectoring mode. As shown in Fig. 1a), we reuse one generic CORDIC block for both modes using a time-serial strategy. The number of iterations can be changed at run-time to trade-off accuracy and latency/throughput.

$$sigmoid(x) = \frac{1}{1+e^{-x}} = \frac{1}{1+\cosh(x)-\sinh(x)} \quad (1)$$

For LUT AF, we implement a small LUT with 16 16-bit entries. A search tree is used to select corresponding entries based on input values, as shown in Fig. 1b). For PWL AF, we use first-order Taylor expansion where a function f(x) can be presented as $f(x) = f(a) + xf'(a) - af'(a) = f'(a)x + (f(a) - af'(a)) = Ax + B$. Coefficients A and B are pre-calculated and stored in two 16-bit register sets. Two entry distributions are investigated in LUT and PWL approaches, as shown in Fig. 2, where entry points are evenly distributed along axis x(Dx) and y(Dy), respectively. For all three approaches, we implement tanh as $tanh(x) = 2sigmoid(2x)-1$ using shift and add operations based on the sigmoid computation flow.

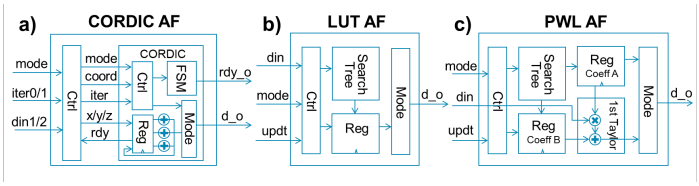

Fig. 1. Block diagrams of CORDIC AF, LUT AF and PWL AF.

This work was supported in part by the Electronic Components and Systems for European Leadership Joint Undertaking ANDANTE under Grant 876925.