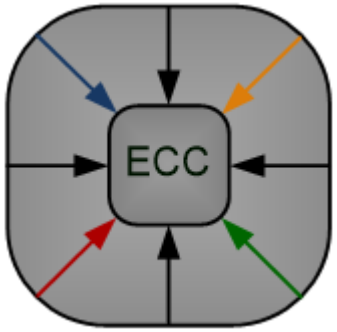
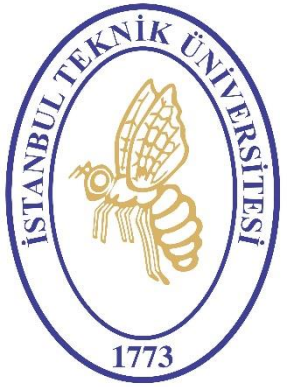
Efficient Time-Multiplexed Realization of Feedforward Artificial Neural Networks



Levent Aksoy, **Sajjad Parvin**, Mohammadreza Esmali Nojehdeh and Mustafa Altun

Emerging Circuits and Computation (ECC) Group

Istanbul Technical University



International Symposium on Circuits and Systems (ISCAS), 10-21 October 2020, Seville Spain

Outline

- Introduction
- Background
- Motivation
- Time-Multiplexed ANN Design
- Experimental Results
- Conclusions

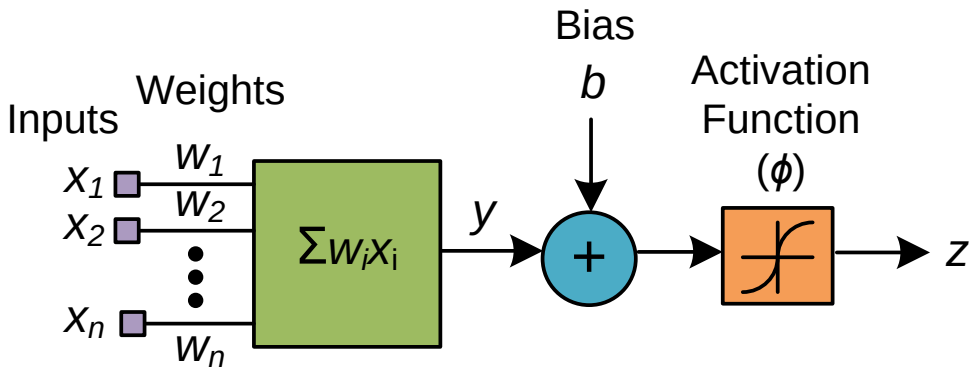
Introduction

- **Artificial neural network** (ANN) is a computing system made up of a number of simple and highly interconnected processing elements
- ANNs have been applied to a wide range of problems
  - classification and pattern recognition
- They have been realized in different design platforms
  - analog, digital, hybrid very large scale integrated (VLSI) circuits, field programmable gate-arrays (FPGAs), and neuro-computers

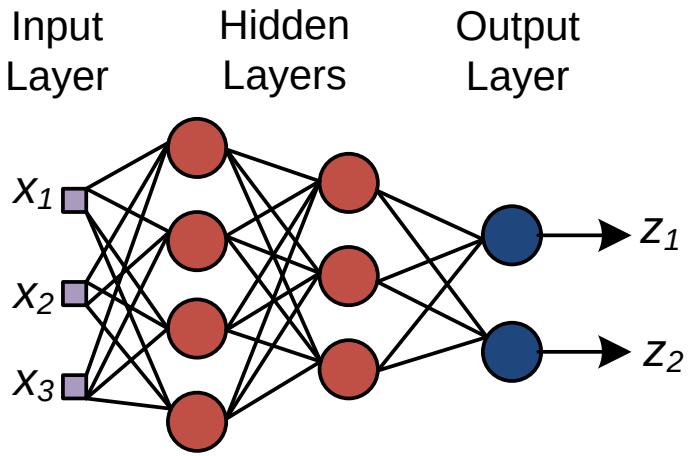
Background

Neuron - a fundamental unit of ANN

$$y = \sum_{i=1}^{n} w_i x_i \qquad z = \emptyset(y + b)$$



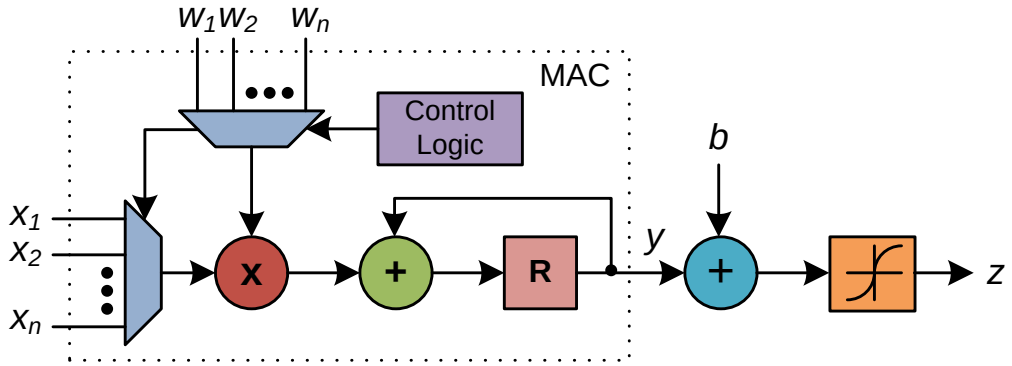
An ANN architecture



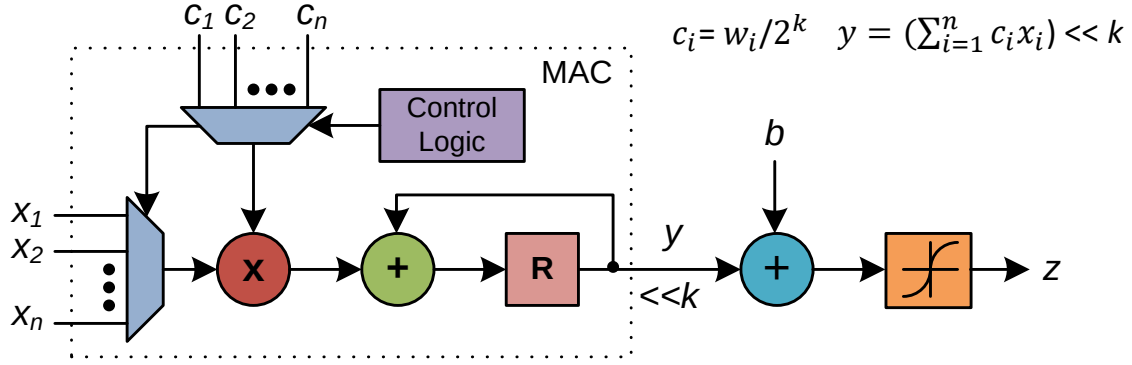
- Hardware complexity of an ANN is dominated by **the multiplication of weights by input variables**.

Motivation

Time-multiplexed design of a neuron



Simplified time-multiplexed design of a neuron

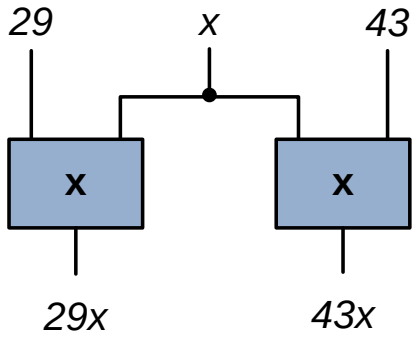


- Control logic – an up-counter
  - Complexity $\propto$ number of inputs (or weights)
- Multiplexers
  - Complexity $\propto$ number and bitwidths of inputs and weights
- Multiplier
  - Complexity $\propto$ maximum bitwidths of inputs and weights
- Adder and register (R)
  - Complexity $\propto$ bit-width of the inner product of inputs and weights

$$y = \sum_{i=1}^{n} w_i x_i$$

Motivation

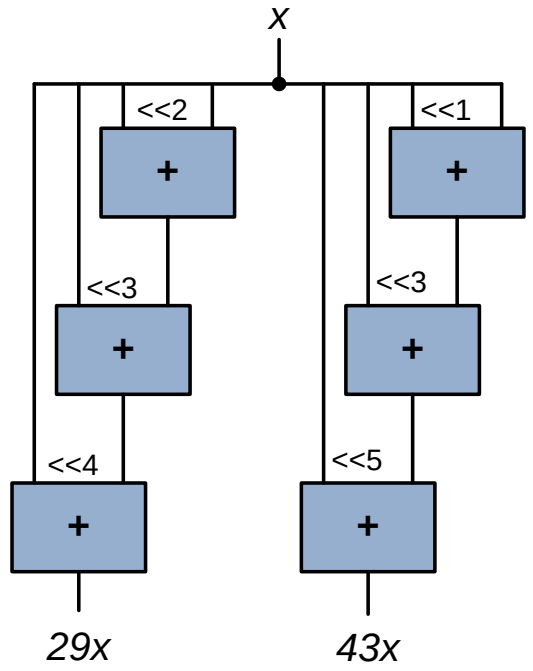
- **Multiple constant multiplications** (MCM) operation realizes the multiplication of multiple constants by an input variable *x*
  - Since constants are determined beforehand, the MCM block can be realized under the **shift-adds** architecture using only **addition, subtraction, and shift** operations



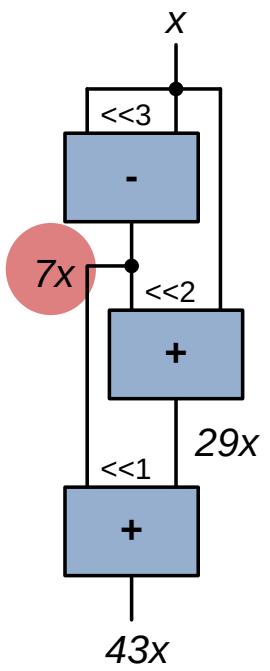
Digit Based Recoding [1]

$29x = (11101)_2x = x\ll4 + x\ll3 + x\ll2 + x$

$43x = (101011)_2x = x\ll5 + x\ll3 + x\ll1 + x$



Exact Method [2]



[1] M. Ercegovac and T. Lang, Digital Arithmetic. Morgan Kaufmann, 2003.

[2] L. Aksoy, E. O. Gunes, and P. Flores, "Search algorithms for the multiple constant multiplications problem: Exact and approximate," Microprocessors and Microsystems, Embedded Hardware Design (MICPRO), vol. 34, no. 5, pp. 151–162, 2010.

Time-Multiplexed ANN Design

- The design procedure has three main steps:
  1) Given the ANN structure, train the ANN using state-of-art techniques and find the weight and bias values
  2) Post-training stage
     a) Determine the minimum quantization value
     b) Convert the floating-point weight and bias values to integers
     c) Tune the weight and bias values such that the time-multiplexed design complexity is reduced
  3) Describe the time-multiplexed ANN design in hardware

Training

- Our training tool includes
  - several iterative optimization algorithms, namely conventional and stochastic gradient descent methods and **Adam optimizer** [3]
  - different weight initialization techniques, namely **Xavier** [4], He [5], and fully random
  - several stopping criteria, namely number of iterations, **early stopping using validation data set**, and saturation of logic functions
  - different activation functions for neurons in each layer, namely **sigmoid**, hyperbolic tangent, hard sigmoid, **hard hyperbolic tangent**, linear rectified linear unit, and softmax

[3] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv e-prints, 2014, arXiv:1412.6980.
[4] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.
[5] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” arXiv e-prints, 2015, arXiv:1502.01852.

Hardware-aware Post-training

- Computing the minimum quantization value
  1) Set the ***quantization value***, *q*, and the related ***ANN accuracy in hardware***, *ha(q)*, to 0
  2) Increase *q* by 1
  3) Convert each floating-point weight and bias value to an integer by multiplying it 2^q and ceiling this multiplication result
  4) Compute *ha(q)* value on the **validation data set** using the integer weight values
  5) If *ha(q)* > 0 and *ha(q)* – *ha(q-1)* > 0.1%, go to Step 2
  6) Otherwise, return *q* as the minimum quantization value

Hardware-aware Post-training

- Tuning the weight and bias values

1) Set *ha(q)* to the **best ANN accuracy in hardware,** *bha*

2) For each neuron in the ANN design, n_i, compute the **smallest left shift** value, *sls*, among its weights

   a) For each weight associated with n_i, wn_ij, find its **largest left shift value**, *lls*

   b) If *lls* is equal to *sls*, determine possible weights as $pw_1 = wn_ij - (wn_ij \bmod 2^{lls+1})$ and $pw_2 = pw_1 + 2^{lls+1}$, compute the ANN accuracy ha_1 and ha_2 when wn_ij is replaced by pw_1 and pw_2, respectively

   c) If $max(ha_1, ha_2) \geq bha$, replace wn_ij by the associated weight and update *bha*

   d) Otherwise, assume wn_ij is replaced by a possible weight with a maximum accuracy, change **bias value**, bn_i in between $[bn_i - 4, bn_i + 4]$ and compute ANN accuracy. If ANN accuracy is equal to or better than *bha*, update the values of wn_ij, bn_i, *bha*

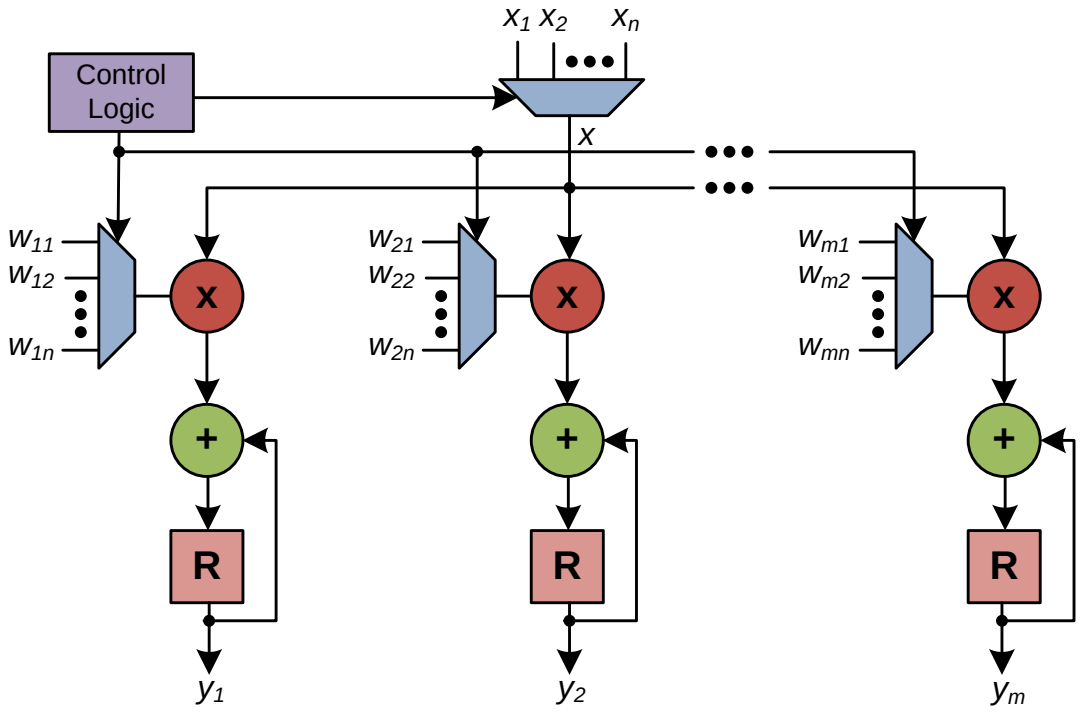
3) If *sls* value of any neuron is improved, go to Step 2

4) Otherwise, return the weight and bias values

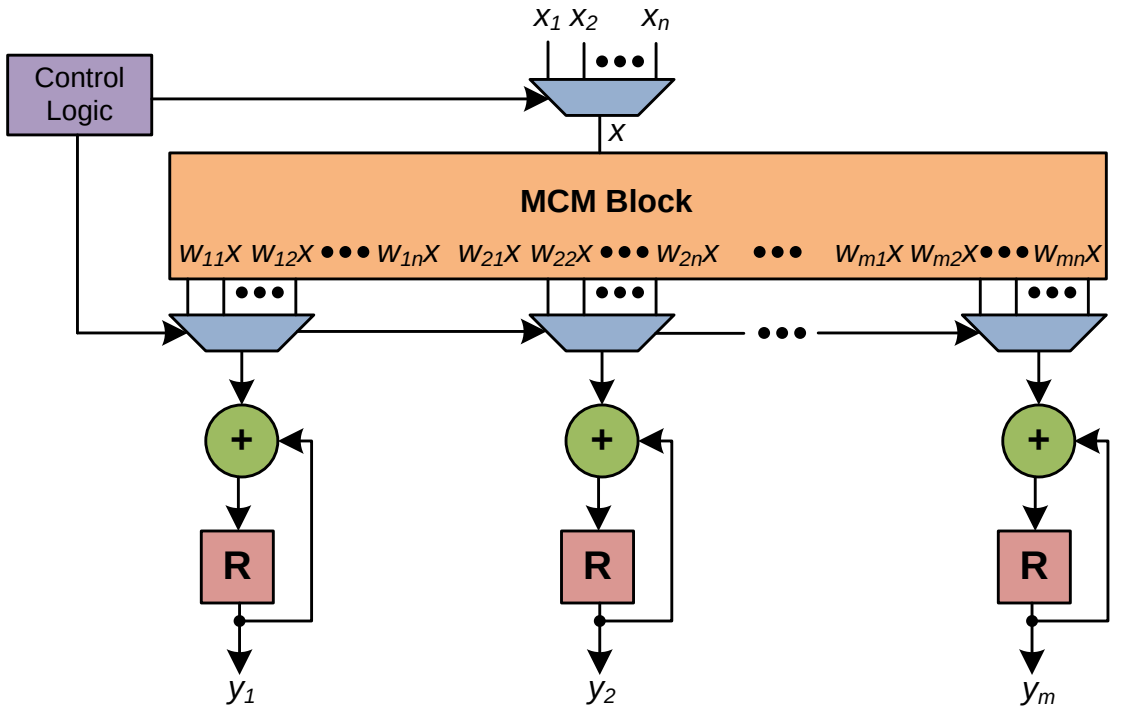
Hardware Design and Multiplierless Design Optimization

- Each layer of the ANN design is implemented using MAC blocks

With Multipliers



Without Multipliers



Experimental Results

- **Pen-based handwritten digit recognition** problem [21] was used as an application.
- In the convolutional neural network design of this application, 5 ANN structures with different number of hidden layers and number of neurons in the hidden layers were used.
- ANN structures were implemented in three different architectures
  - Parallel
  - Time-multiplexed using multipliers – TM-MUL
  - Time-multiplexed using multiplierless MCM blocks – TM-MCM
- ANN designs were described in **Verilog** and synthesized using the **Cadence RTL Compiler** with the **TSMC 40nm design library.**

Experimental Results

ANN Design Results without Post-Training

Structure	Training Details		Parallel				TM-MUL				TM-MCM				
	sta	hta	area (μm^2)	latency (ns)	power (mW)	energy (pJ)	area (μm^2)	latency (ns)	power (mW)	energy (pJ)	add	area (μm^2)	latency (ns)	power (mW)	energy (pJ)
16-10	84.6	83.3	19779	3.3	8.6	28.4	6098	55.0	0.6	35.4	110	9222	63.5	0.9	57.7
16-10-10	94.1	92.9	30765	6.2	15.8	98.4	11682	96.7	1.2	113.9	169	15854	110.7	1.2	128.4
16-16-10	96.0	94.7	42464	6.5	24.0	155.7	14301	127.3	1.5	186.3	212	18941	148.4	1.4	213.4
16-10-10-10	94.7	91.9	39952	9.0	26.1	235.2	16697	145.3	1.6	226.8	216	19909	153.4	1.1	172.1
16-16-10-10	96.6	94.6	50737	7.3	28.7	209.3	19100	168.3	1.7	293.1	232	23001	183.3	1.2	228.4
Average	93.2	91.5	36739.4	6.5	20.6	145.4	13575.6	118.5	1.3	171.1	187.8	17385.4	131.9	1.2	160.0

ANN Design Results with Post-Training

Structure	Post-Training Details		Parallel				TM-MUL				TM-MCM				
	q	hta	area (μm^2)	latency (ns)	power (mW)	energy (pJ)	area (μm^2)	latency (ns)	power (mW)	energy (pJ)	add	area (μm^2)	latency (ns)	power (mW)	energy (pJ)
16-10	5	86.6	8943	2.8	2.7	7.4	4278	47.2	0.4	19.0	11	3633	47.8	0.4	17.8
16-10-10	7	93.5	16838	5.4	8.6	46.0	8676	85.8	0.9	76.1	34	7648	91.5	0.6	57.5
16-16-10	7	95.9	20063	5.5	10.7	58.6	10014	114.0	1.0	112.3	33	8358	122.4	0.8	101.2
16-10-10-10	7	93.5	22994	7.9	13.7	108.4	13020	121.2	1.2	150.1	43	10743	134.6	1.0	132.1
16-16-10-10	7	95.6	25181	5.5	13.2	73.2	13368	147.8	1.3	199.3	29	11033	152.4	1.0	152.1
Average	6.6	93.0	18803.8	5.4	9.8	58.7	9871.2	103.2	1.0	111.4	30.0	8283.0	109.7	0.8	92.1

Conclusions

- This paper presented efficient techniques to reduce the hardware complexity of a time-multiplexed feedforward ANN design
  - a post-training stage where the size and values of constant weights are explored to reduce the hardware complexity
  - the multiplierless realization of the time-multiplexed ANN design
- It is shown that the proposed techniques yield a significant reduction in design complexity

Questions

THANKS for YOUR ATTENTION

Contact: Sajjad Parvin

E-mail: sajadparvin.stu93@gmail.com