5-2018

Configurable Low Power Analog Multilayer Perceptron

Jeffery M. Dix
University of Tennessee

To the Graduate Council:

I am submitting herewith a dissertation written by Jeffery M. Dix entitled "Configurable Low Power Analog Multilayer Perceptron." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Electrical Engineering.

Benjamin J. Blalock, Major Professor

We have read this dissertation and recommend its acceptance:

Vasilios Alexiades, Jeremy H. Holleman III, Lynne E. Parker

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Configurable Low Power Analog Multilayer Perceptron

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Jeffery M. Dix

May 2018

To my Mom and Dad: you always pushed me to surpass my limits and test my boundaries. I would not have made it this far without your support and love.

Acknowledgements

I would like to thank Dr. Blalock, my lab colleagues, and ICASL for providing me with a great environment to learn and develop analog circuits. I would also like to thank Dr. Holleman and his graduate students for their knowledge and support during the development of my PhD work. Lastly, I would like to thank the University of Tennessee and all of its wonderful professors for the enjoyable and supportive culture that is in place in Knoxville.

Engineers like to solve problems. If there are no problems handily available, they will create their own problems. -Scott Adams

Abstract

A configurable, low power analog implementation of a multilayer perceptron (MLP) is presented in this work. It features a highly programmable system that allows the user to create a MLP neural network design of their choosing. In addition to the configurability, this neural network provides the ability of low power operation via analog circuitry in its neurons. The main MLP system is made up of 12 neurons that can be configurable to any number of layers and neurons per layer until all available resources are utilized. The MLP network is fabricated in a standard 0.13 μm CMOS process occupying approximately 1 mm^2 of on-chip area. The MLP system is analyzed at several different configurations with all achieving a greater than 1 Tera-operations per second per Watt figure of merit. This work offers a high speed, low power, and scalable alternative to digital configurable neural networks.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Artificial neural networks (ANNs) are networks that are used for machine learning and are inspired by biological neural structures (i.e. the brains of animals). ANNs attempt to mimic these biological neural structures in how they function. For example, an ANN can be used to approximate a function that is dependent upon a large set of generally unknown inputs [26]. There exist many different types of neural network designs with each being highly dependent upon how data is to be interpreted. For the case of a supervised learning structure with perceptrons [26], a single layer perceptron interprets data linearly whereas the multilayer perceptron (MLP) has the ability to arbitrarily classify a set of data with nonlinear functions [1]. Multilayer perceptrons are advantageous structures because of their ability to compute in parallel several different inputs, which are then passed through their overall algorithm as quickly as possible.

Multilayer perceptrons are a type of artificial neural network that perform computations on a set of data using simple units with weighted connections. The structure of a MLP, in its most basic form, consists of an input layer, a hidden layer, and an output layer. The input layer takes the set of data to be analyzed and passes it on to the hidden layer(s). The hidden layer may be made up of one or more layers that take the outputs from the previous layer, multiply them with a weight, and output this weighted value via a nonlinear activation function. Each layer consists of one or more neurons that create the overall ANN functionality. Two of the common nonlinear activation functions are the sigmoid (equation 1.2) and the hyperbolic tangent (equation 1.3) functions with the general form shown in equation 1.1.

$$y_i(\mathbf{x}) = \phi(\mathbf{w}_i'\mathbf{x} + b_i) = \phi(z_i) \tag{1.1}$$

where $\mathbf{x}$ is the vector of input values, $\phi()$ is the nonlinear training function, $\mathbf{w}_i'$ is the vector containing the weight values, and b_i is the bias for a given neuron.

$$y_i(\mathbf{x}) = \frac{1}{1 + \exp(-z_i)} \tag{1.2}$$

$$y_i(\mathbf{x}) = \frac{\exp(z_i) - \exp(-z_i)}{\exp(z_i) + \exp(-z_i)} \tag{1.3}$$

The output layer receives the final hidden layer's output, multiplies that output with another weight, and then again passes the new value through a non-linear function to produce the final output. Figure 1.1 shows a MLP with two hidden layers. A distinct characteristic of a MLP ANN is that every neuron will output to all of the neurons of the subsequent layer [1].

Typically, the hidden and output layers are constructed of the same signal pathway: inputs from previous layer being weighted, then summed with the other weighted inputs, and finally passed through an activation function that will either output the final target value or pass the output on to the next layer neurons. Figure 1.2 details a simple block diagram of the arithmetic in a node within a MLP. In a MLP, the weights are adjusted by training the overall ANN to have as little error as possible between its final outputs and assumed final target values. Some standard training criteria consist of least squares error and cross-entropy as well as other algorithms depending upon a designer's constraints [1]. Through training of the MLP, the ANN provides outputs that are close to the theoretical values that can be obtained mathematically. A standard MLP output can take the form of the following:

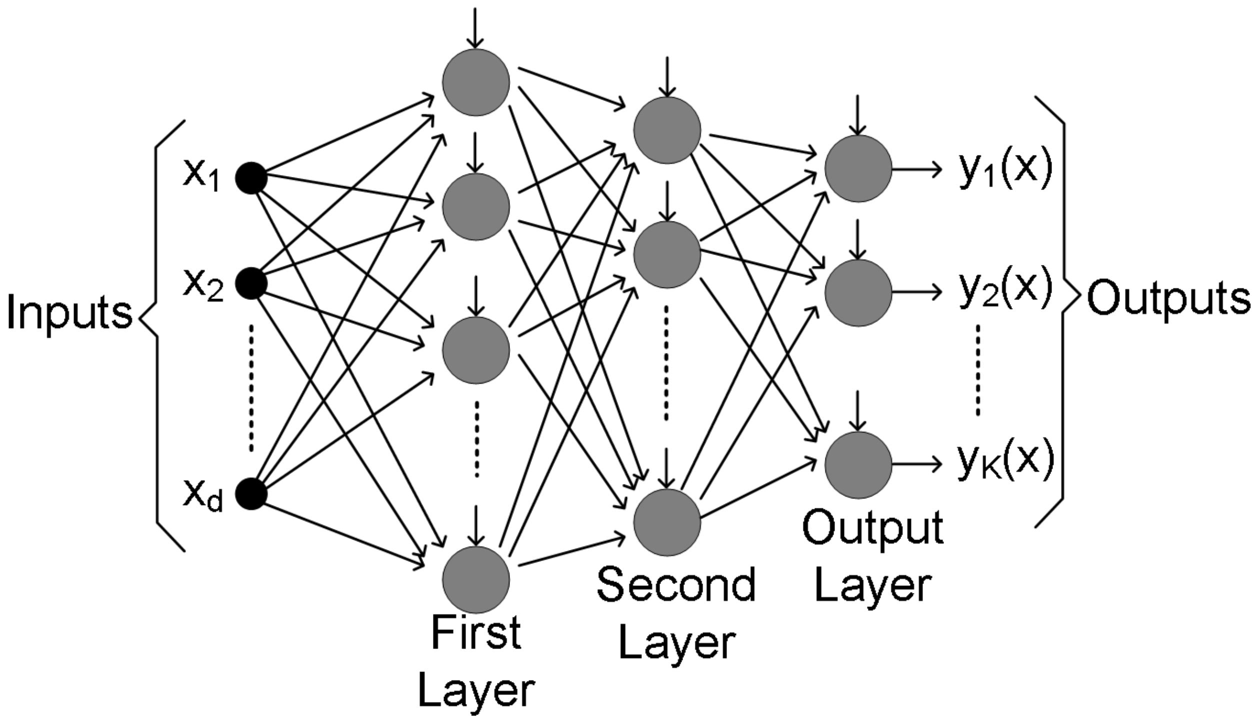


Figure 1.1: Two Hidden Layer MLP Example.

$$y_i(x^k) = \phi(\sum_{j=1}^{n} w_{ij} x_j^k + b_i) \tag{1.4}$$

where $\phi()$ represents the nonlinear activation function, w_{ij} represents the training weight, $x_j(k)$ represents the input value, and b_i represents the bias for a given neuron so that a neuron's output behaves as expected and is not saturated towards a high or low state [1].

Multilayer perceptrons are a useful type of artificial neural network that has been thoroughly developed for the past few decades in terms of complexity and capabilities. A typical use for a MLP is the classification of a set of data that is not linearly separable [1]. The classification of the data allows a complex set of data with a high number of dimensions to be broken down into a simpler set of data with fewer dimensions. The MLP is a versatile construct but can also be implemented in such a fashion that the network is too complex or consumes too much operational power to be useful in certain applications. A general MLP structure with its base neuron elements can be seen in Figure 1.3.

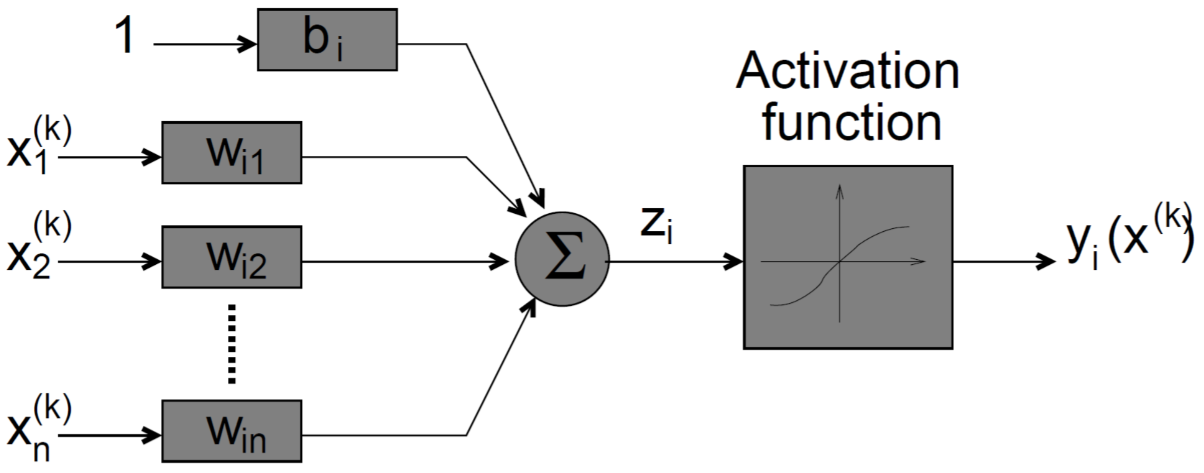


Figure 1.2: MLP Basic Neuron Block Diagram [1].

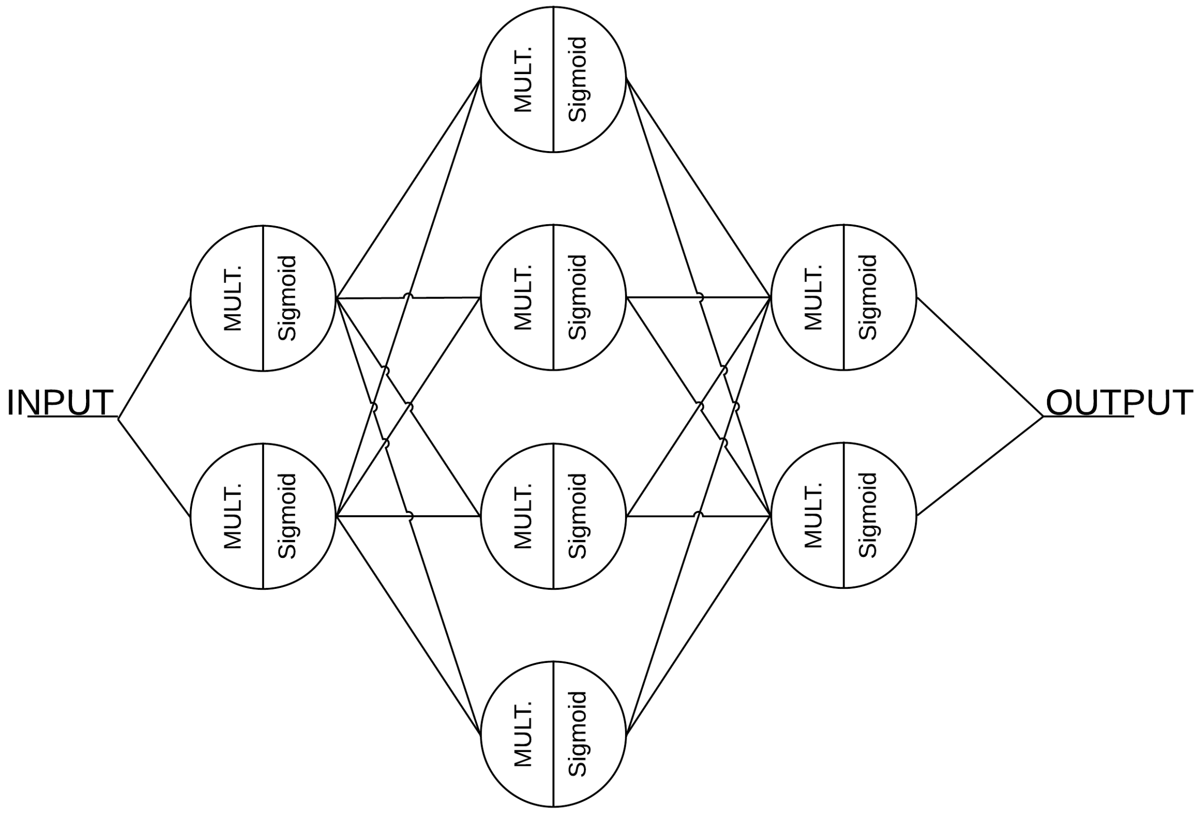


Figure 1.3: General MLP Structure.

1.1.1 Motivation

The complexity and power consumption of a MLP are critical performance characteristics that need to be optimized in order to produce the most efficient neural network possible for a particular problem. The complexity of a MLP can be addressed in many ways. The most popular method among designers currently is utilizing a field programmable gate array (FPGA) to easily and efficiently change the architecture of the MLP seamlessly. While this digital hardware based method provides the configurability that designers want, the FPGA does not efficiently optimize power consumption at the transistor level and can lead to a higher power use for the overall design. Another method to add configurability to a MLP neural network is to manually design circuit blocks on an integrated circuit (IC). This method can provide nearly the same performance characteristics as a FPGA but with greater transparency and control over every device in the network [27].

While the configurability of a custom IC will be limited by the total number of blocks in the overall design, the IC can boast better performance characteristics by leveraging focused design techniques to limit travel time/distance for important signals between MLP nodes. Overall, with these two methods in mind, configurability of a MLP ANN is a key design parameter that allows the network as a whole to function in an optimal manner with the ability to turn off and on sections of the ANN without penalty (and in most cases some benefit) to the flow of data. Even further, the data analysis can greatly benefit from a configurable design that can break a large data stream apart and operate simultaneously on different sections of the data at the same time. This operability could greatly reduce the overall computation time or increase the data transmission rate in the ANN.

After considering the configurability of a MLP, the overall power consumption of a particular network needs to be ascertained. The goal of the ANN is to provide the highest number of computations per second per Watt of power. The tradeoff in this case is that generally the more power one can supply a MLP ANN the more computations per second it should be able to perform. However, the downside of giving a MLP more power is that you then limit the applications that the ANN would be useful in implementing since it now requires a greater power and therefore a larger system built around it. Limiting the power gives a way for the MLP to be useful in applications away from constant power supplies. Even with a constant power supply, a power limit is useful enhancing application mobility, lowering environmental impact, or even cost to the end user.

With the idea of limiting power while optimizing performance, a MLP designer is required to move away from digital structures and utilize analog constructs. Analog designs as a whole have proven over time that they are far more capable of low power computations (e.g., [28] and [29]). Analog allows the designer to control power dissipation at a transistor level while also taking advantage of transistor characteristics to optimize performance for a particular power level. Analog circuits are also capable of functioning with nearly the same efficiency in data analysis as their digital counterparts. In terms of data analysis, digital circuits are more efficient when a part of a larger digital system that will feed the ANN digital inputs and expect digital outputs. On the other hand, an analog ANN can directly process signals from the world without bulky, power consuming analog-to-digital (ADCs) in the IC. Another power saving aspect of analog circuits is that summation and subtraction operations are easily performed by a wire junction (current-mode circuits) whereas digital circuits require multibit subtractors and adders [30].

As mentioned above, the designer can control individual transistor biases in analog circuits such that a transistor will operate in weak inversion. The ability to control the transistor's operation is crucial in creating an ANN that has the basic functionality of a MLP with the ability to consume as low power as possible. Weak inversion provides these capabilities in two ways: exponential transfer characteristics and low power consumption [3]. Exponential transfer characteristics are vital to the successful operation of a MLP. The MLP's neurons, or nodes, require an activation function after the weight adjustments. In the case of MLPs, the activation function is generally a nonlinear operation and requires special circuit designs in order to properly produce the desired nonlinear functions typically seen [31]. As far as low power consumption of weak inversion circuits, this characteristic is attractive to a designer, because low power operation opens up new application spaces for MLPs. With these characteristics in mind, an analog low power MLP design is highly desirable in that it has the greatest potential to provide the highest number of computations per second for the lowest power consumption.

1.2 Problem Statement

As mentioned in the previous section, MLP networks have the advantage of being constructed with either a digital or analog "backbone". With respect to the digital "backbone', the typical design procedure for a configurable MLP network is to utilize a FPGA and software techniques to create an easily manipulated system. The prominent issues with using a FPGA to build configurable MLPs exist in resource and power management. FPGAs are complex systems with thousands upon thousands of devices that can be linked together to build almost anything that can be put in code. Considering this advantage of FPGAs, the configurable MLP network can easily be implemented. Now, this newly formed configurable network does not achieve either of those goals considering the two prominent issues mentioned. In terms of resource management, a designer could develop a MLP in a FPGA that utilizes 100% of all of the resources and has outstanding performance. By following this route, the MLP network will have a high power consumption and therefore a poor figure of merit. Inversely, the designer could develop a MLP system on the FPGA that has good power management techniques but does not use all the resources available. The resource management in this case is drastically reduced resulting in large chunks of area of the FPGA to be unused and wasted.

As for analog-based configurable MLPs, they are inherently built to have good power management through their analog core components. However, the resource management may be lacking in these designs. As shown in the next chapter, current analog configurable MLP designs take one of two methodologies when approaching the configurability of the network. Firstly, the analog network can shut off the unused portions of the overall system leaving only minor leakage pathways to consume some power. Secondly, the analog network could fold the unneeded nodes into the hidden or output layers as extras. The first design choice results in some power savings but completely wastes resources in the analog system. The second design choice does not save any more power and continues to use the unneeded nodes leading to a failure of power and resource management.

Therefore, there exists a gap in design methodologies that can effectively manage power and resources such that power can be optimized to provide an advantageous figure of merit while fully utilizing the available resources in a manner that can add greater capability to the overall system.

1.3 Original Contributions

This work plans to include and demonstrate several contributions to the state of the art for analog, low power MLP systems. The first contribution is a high speed data rate while maintaining a small form factor for the base circuits within the system. An overall configurable system architecture is designed, implemented, and tested in this work. Included in the configurability is the ability to control the biasing structures for each individual cell in the integrated circuit. The next demonstrated contribution will be a low power analog system design so that the system is capable of mobile or power limited applications while maintaining a relatively high throughput. The final contribution that will be demonstrated is the scalable nature of the system such that it can be increased or decreased in size to accommodate a particular set of design parameters for a neural network system.

1.4 Dissertation Overview

Chapter 2 of this dissertation presents a review of the literature as it pertains to the MLP as a whole as well as the individual subcomponents that make up the structure of the system. These subcomponents consist of multipliers, sigmoid functions, switches, and winner-take-all circuitry. Chapter 2 also discusses the key differences between digital and analog MLP systems. Chapter 3 details the individual subcomponent designs that are used in this project. Chapter 4 describes key simulation results for each subcircuit along with the system simulations in addition to the test setup used in obtaining the measurement results that are also presented in this chapter. Chapter 5 provides conculsions from the measurement results and design process, and also discusses avenues for future work that can be performed.

Chapter 2

Literature Review

2.1 Multilayer Perceptron Review

As discussed in the previous section, multilayer perceptrons are versatile networks that are useful in a wide range of applications from business to medical fields to pattern recognition to driving [32]. Figure 2.1 shows an example application of a MLP in the field of autonomous driving [2]. MLPs are typically constructed in two distinct ways by being composed of either digital circuits (via FPGAs and coding or an IC consisting of digital circuits) or analog circuits (IC consisting of analog-based arithmetic processes). Analog MLPs are well defined and have a plethora of literature supporting their development for a designer to draw design methodologies. The primary focus in this work will be on the analog MLP structure with a focus on subthreshold (weak inversion) operation at the transistor level and configurable circuits.

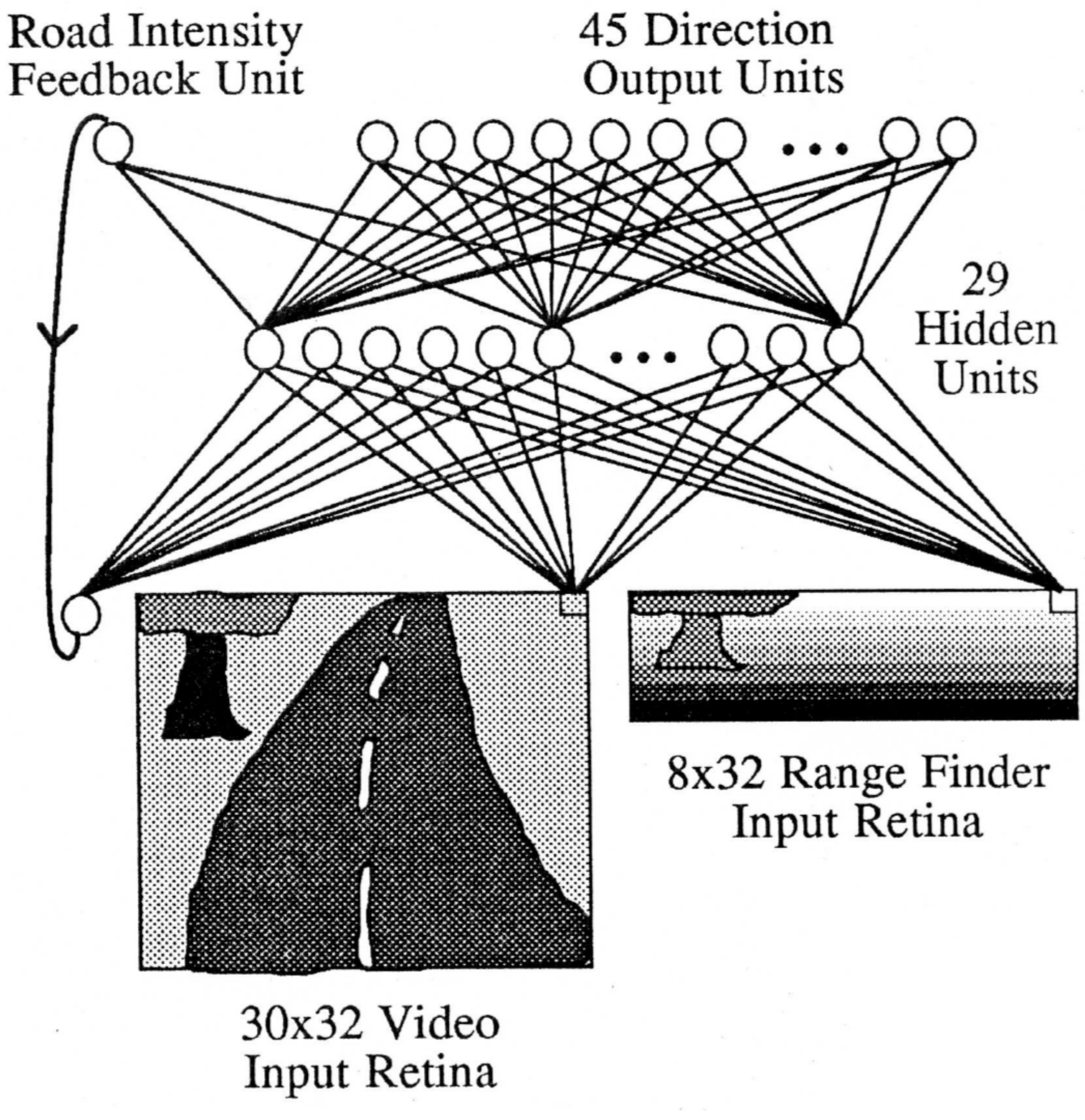


Figure 2.1: ALVINN Autonomous Driving MLP Example [2].

The analog MLP presented in [3] was designed by the authors in order to provide a greater synaptic density while exploiting nonlinear synapses for use with an error back-propagation algorithm. The authors utilized a 3-μm CMOS process to fabricate their MLP design. The design took advantage of analog differential input pairs and current to voltage conversion techniques to minimize the area and therefore increase the overall density on the IC. Figure 2.2 depicts the authors' block diagram scheme for how inputs enter their system and propagate through their neurons being changed by the various computational blocks along the way. However, the authors bias their transistor networks in strong inversion leading to an overall power consumption of 25 mW.

The authors of [4] present an analog MLP architecture that was developed in AMS CMOS 0.35 μm with a supply voltage of 3.3 V. Their test device consisted of a single neuron layout constructed of three multipliers, an adder, and a hyperbolic tangent cell. These structures were implemented in an analog format to utilize differential current inputs being relayed by current mirrors to the necessary evaluating circuitry. Figure 2.3 details the block schematic of the ANN structure. The authors took advantage of analog inputs from measurement devices in order to avoid digital-to-analog conversions on the frontend. The overall power consumption of this design was approximately 49.5 mW.

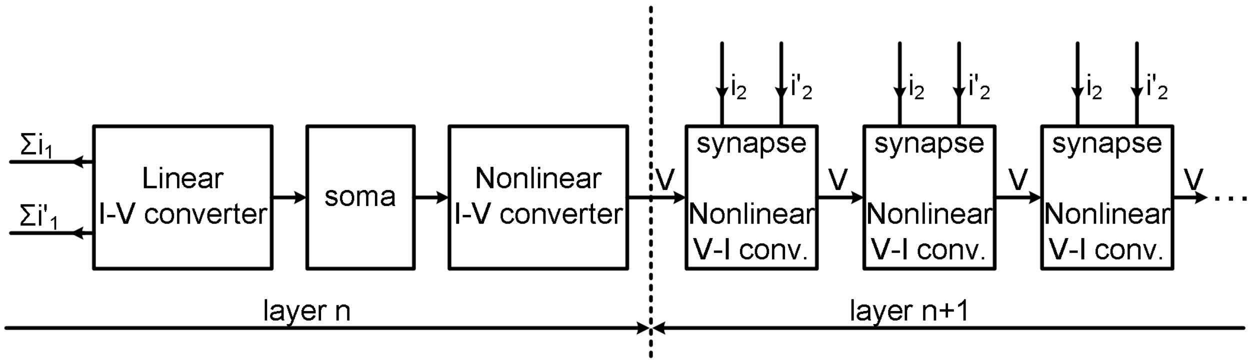


Figure 2.2: Block Diagram of MLP Implemented in [3].

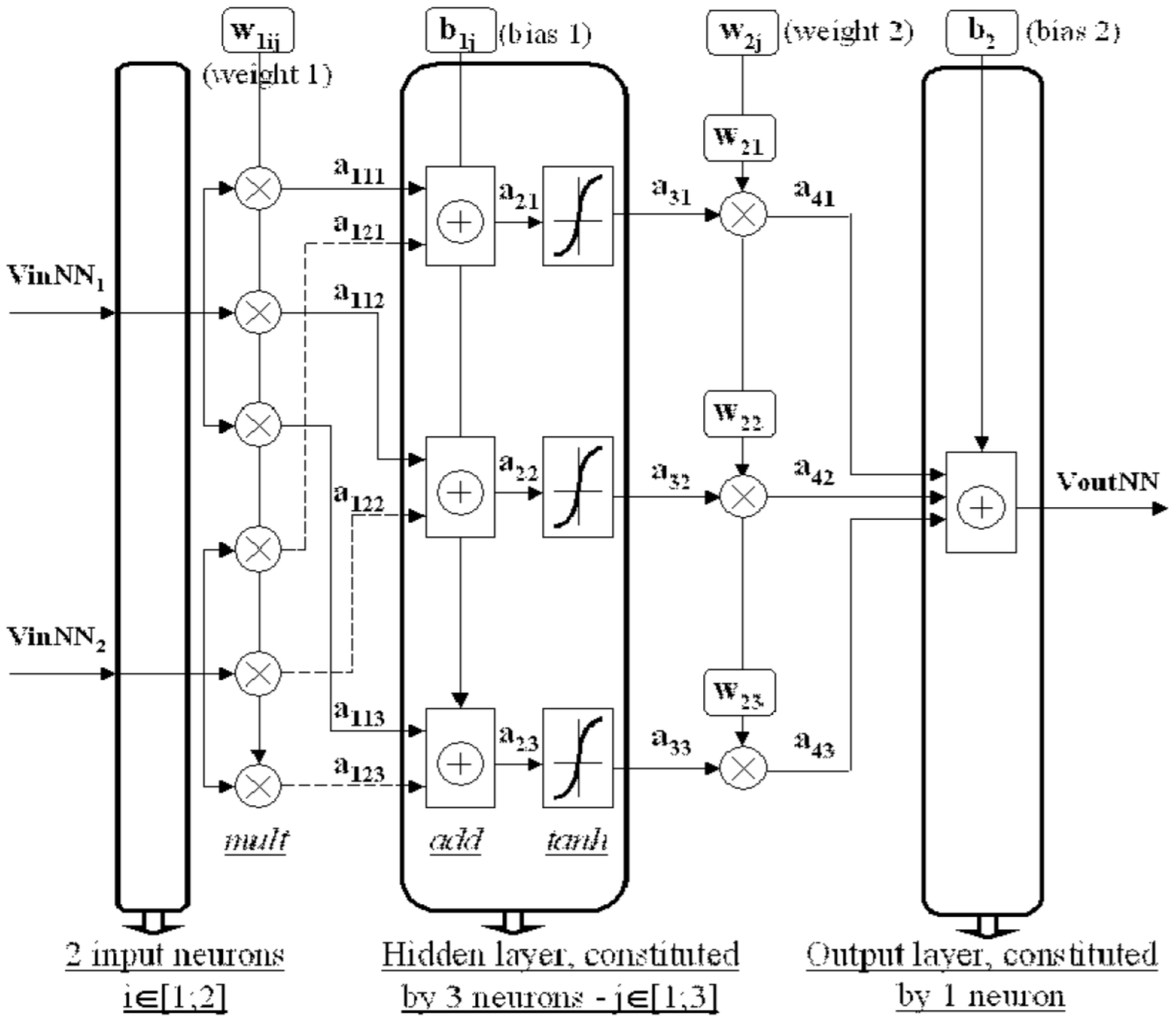


Figure 2.3: ANN Block Schematic from [4].

Diotalevi *et al.* present a generic MLP neural network geared towards low power and low voltage circuitry as well as increased scalability for any future network designs [5]. The authors reported a power consumption of 3 μW as well as 30 MCPS/synapse (Mega-Connections per second per synapse). The low power consumption was achieved utilizing transistors in weak inversion with low bias currents ($\sim$500 nA reported maximum). Again, the MLP used differential currents to connect the input, hidden, and output nodes and was fabricated in an AMS CMOS 0.8 μm process. Figure 2.4 shows the flow of the MLP. The transconductors converted the input signals to differential currents while the synapse contains the circuitry to convert the weight voltages to currents and multiply those with the input currents. The neuron cells implement the nonlinear activation function for this MLP, which is a hyperbolic tangent in this case.

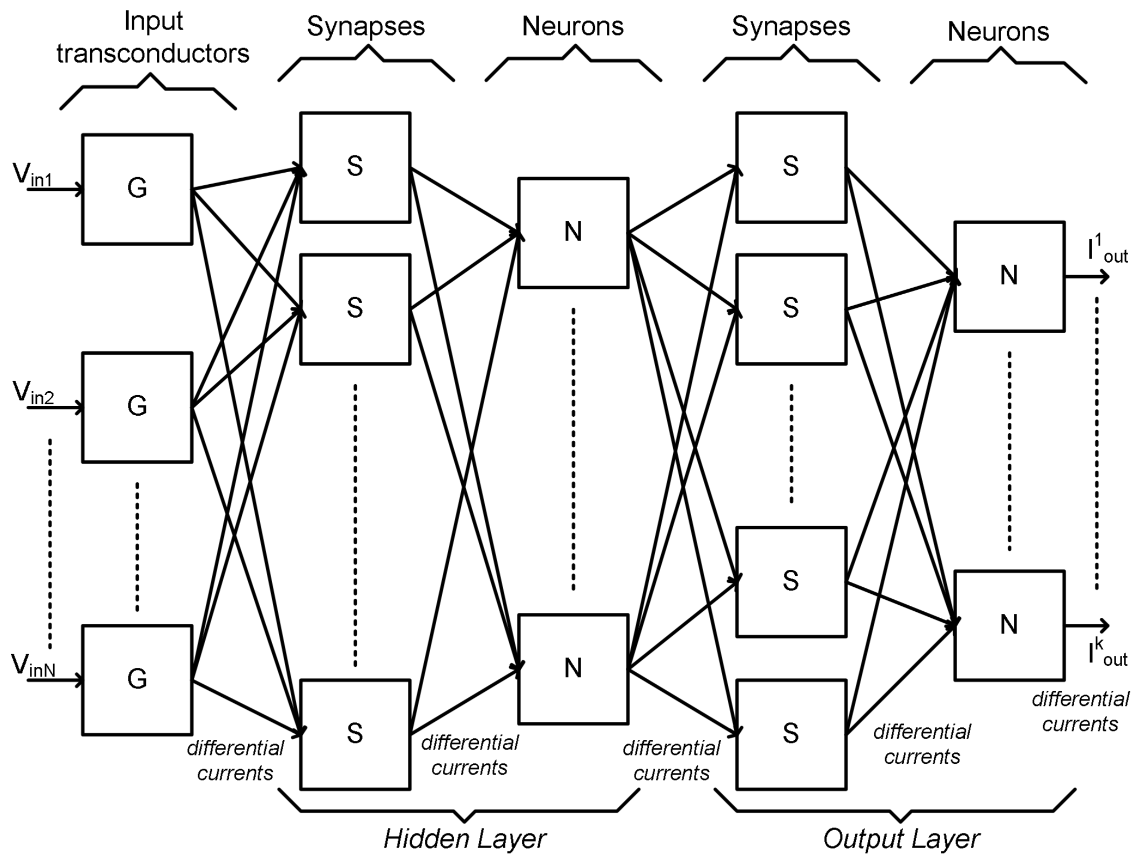


Figure 2.4: MLP Utilizing Current Mode Analog with Differential Currents [5].

Gatet *et al.* detail a different approach to designing an analog MLP in [33]. While their goal is still to minimize power consumption and area, they also aim to achieve a high bandwidth in order to allow for faster data processing. The authors report a power consumption of 595 mW with a bandwidth of 240 MHz in a 0.6 μm CMOS process. The higher power consumption is seen in the multiplier-adder cell the authors developed for increased bandwidth as well as the 5 V power rail. They use the higher voltage to increase the headroom in their multiplier-adder cells with a hyperbolic tangent circuit load to achieve their desired bandwidth.

Bo *et al.* present an analog MLP designed in ATMEL CMOS 0.7 μm process that has a reported simulated power consumption of 25 mW and an energy efficiency of 40 MCPS/mW [34]. The chip designed by Bo *et al.* focused more on implementing a self-learning architecture in an analog design than limiting power consumption. Maliuk and Makris describe an adaptive neural network design with core circuitry biased in weak inversion [6]. The authors achieve their adaptive network through row and column control circuitry adding complexity and power consumption to the overall network design. However, they utilized a floating gate device [6] in a PMOS transistor to store the weights for their learning algorithm on-chip. Figure 2.5 details the floating gate PMOS transistor. The voltage provided by this storage cell is then applied to diode-connected MOSFETs providing a bias voltage to subsequent current mirrors.

Bo *et al.* detail an integrated circuit that is capable of around 2.5 GCPS with an overall power consumption of 200 mW [7]. The higher power consumption comes from the large number of synapses used in the overall architecture (4810 synapses with $\sim$150,000 transistors). The authors further quantify their reported results with a processing delay of 2 μs and about 80 pJ of energy per connection in an ES2 CMOS 1.0 μm process. Figure 2.6 depicts the block diagram specific to this MLP design. Furthermore, the chip takes advantage of transistors biased in weak inversion as well as a current mode approach similar to previously discussed designs.

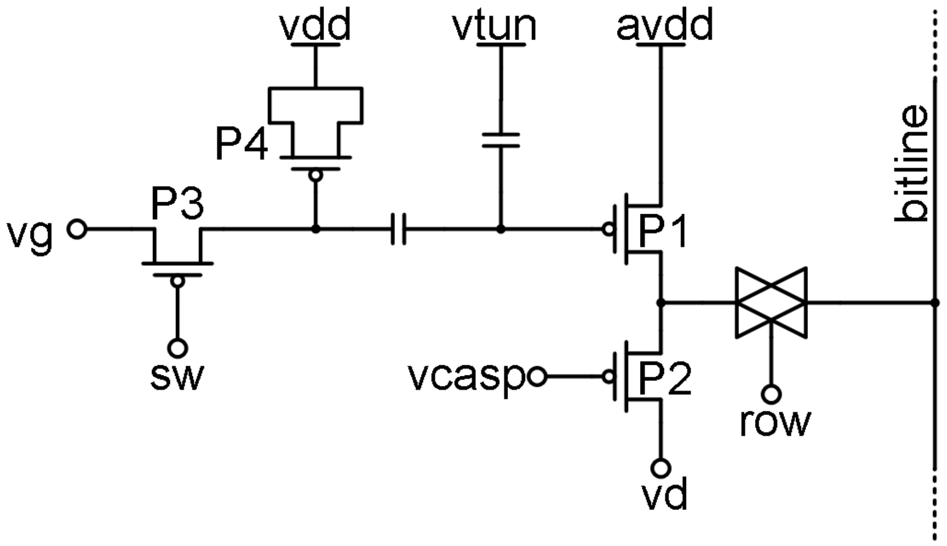


Figure 2.5: Floating Gate PMOS Transistor Implemented in [6].

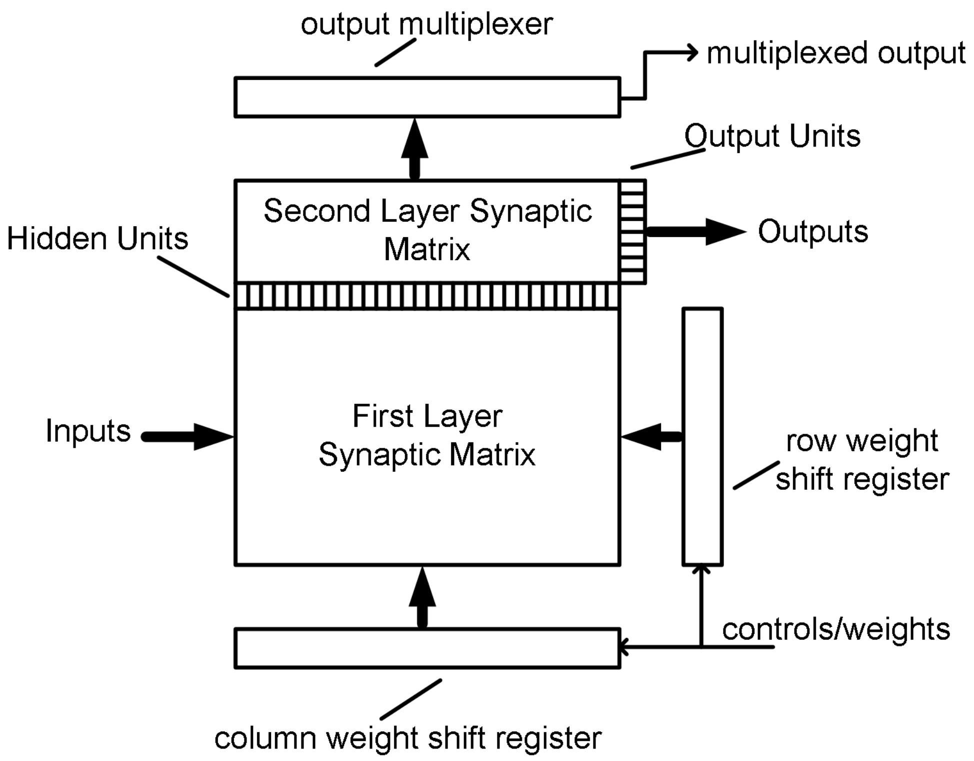

output multiplexer
multiplexed output
Output Units
Second Layer Synaptic Matrix
Hidden Units
Outputs
First Layer Synaptic Matrix
Inputs
row weight shift register
controls/weights
column weight shift register

Figure 2.6: Block Diagram of MLP Constructed in [7].

Instead of only developing a MLP for a specific application, Cairns and Tarassenko present relevant issues with on-chip learning in analog MLPs [35]. They detail that overall an analog MLP can achieve the same performance in terms of accuracy, but not necessarily speed, as that of a software back-propagation technique. They conclude that an effective learning strategy specific to hardware must be implemented. In addition to analyzing training techniques, the authors also discuss weight perturbation techniques and how the need for precise weight memory is vital to the overall operation of the neural network.

Silva *et al.* propose a reconfigurable MLP architecture that was implemented in a FPGA in [8]. The authors present a unique design that allows hundreds of neurons to exist per layer as well as a seemingly infinite number of layers depending on how hardware is synthesized in the FPGA. Furthermore, the authors search for simple arithmetic circuits that would require less physical area to perform functions on real numbers leading to the use of fractions to represent these real numbers. All hardware implementations are created using digital circuits on the FPGA that lead to a heavy number of resources being utilized. Though the overall power consumption is not reported, it can be inferred from the number of resources used that power would most likely be on the order of milli-Watts or greater. Figure 2.7 depicts the amount of resources used to build a neuron circuit that is digitally based in a FPGA. Though the digital circuits can handle higher frequencies of data, any power-based figure of merit is weakened considerably by power consumption due to the digital design.

Su *et al.* present an adaptive analog MLP hardware implementation that uses bipolar transistors instead of MOSFETs for all functions in [9]. Figure 2.8 shows two examples of the bipolar transistor circuits used in this design. The authors state that a bipolar differential pair can perfectly generate a hyperbolic tangent function. This function can be seen in one of the outputs of Figure 2.8a and by:

$$I_{C1} = \frac{I_0}{2}\left(\tanh\left(\frac{V_{in}}{V_T}\right) + 1\right) \tag{2.1}$$

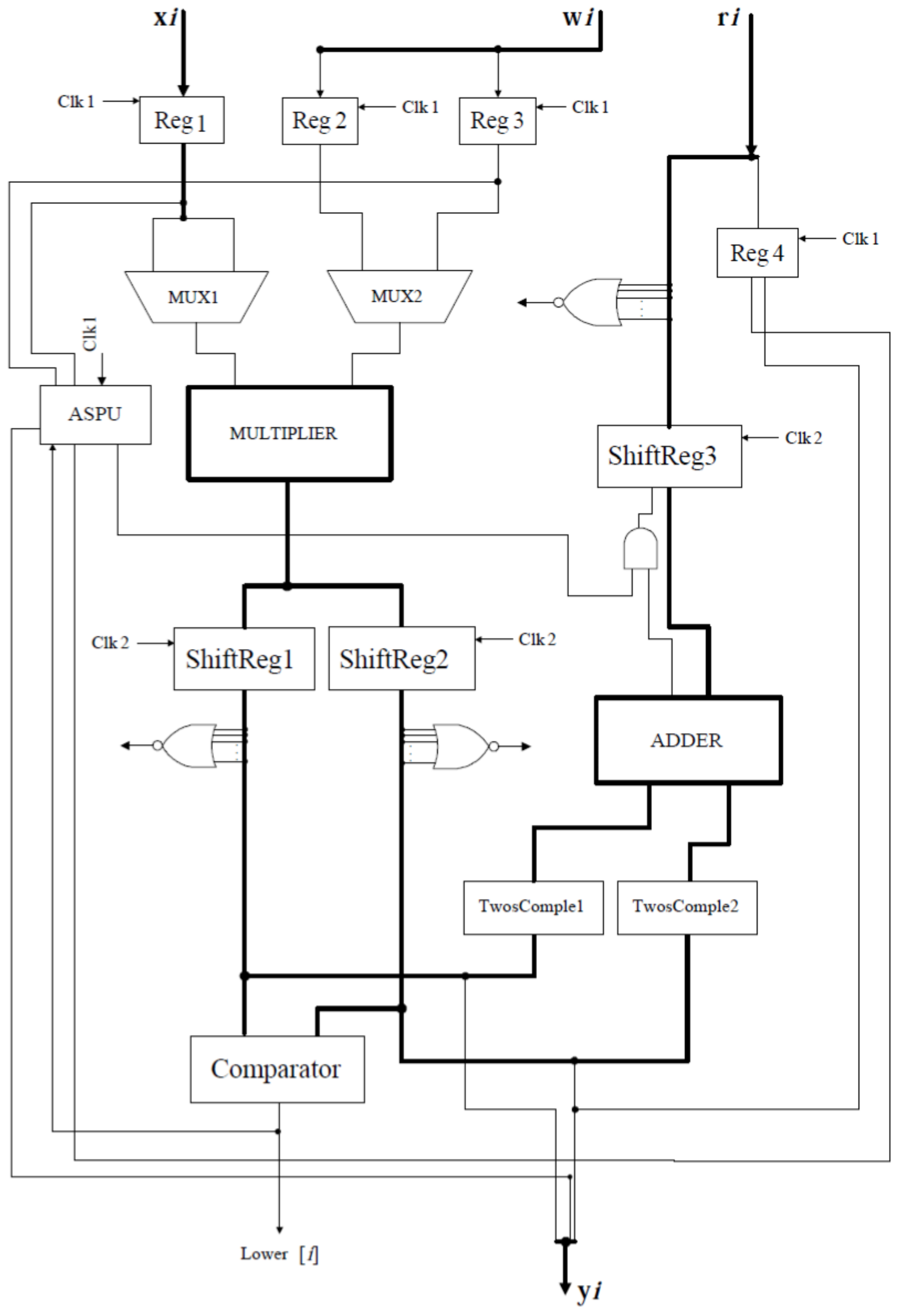


Figure 2.7: Digital Example of a Neuron Circuit in a FPGA [8].

where I_{C1} is the collector current in the first transistor of the differential pair, I_0 is the tail current for the bipolar transistor differential pair, V_{in} is the differential input voltage, and V_T is the thermal voltage (0.025 V at room temperature). To further refine the tanh function, Su *et al.* add an active current mirror load to the differential pair, which is seen in Figure 2.8b and creates a more refined hyperbolic tangent function given in Equation 2.2.

$$I_{diff,out} = I_0 \tanh\left(\frac{V_{in}}{2V_T}\right) \tag{2.2}$$

Furthermore, the authors propose that the bipolar transistors allow the neural network to operate in the GigaHertz range while also providing some adaptive capabilities and better cost efficiency. The adaptive capabilities consist of varying an input current through outside controllers for differential control applications or nonlinear model systems. The authors report an error around 5% but do not report the total power consumption or speed of the overall neural network. Figure 2.9 details the block diagram for this design. The block diagram shows that the authors took particular care in developing a system that can operate in a bidirectional fashion by developing a system that can operate with negative currents (absolute and sign blocks rectify the necessary signals) instead of focusing on power consumption or configurability.

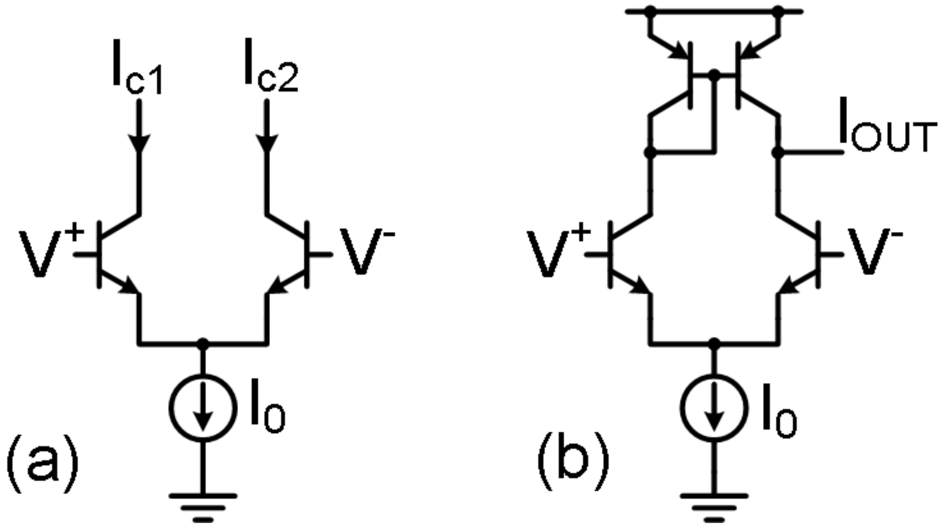


Figure 2.8: Differential Pair with No Load (a) and Active Current Mirror Load (b).

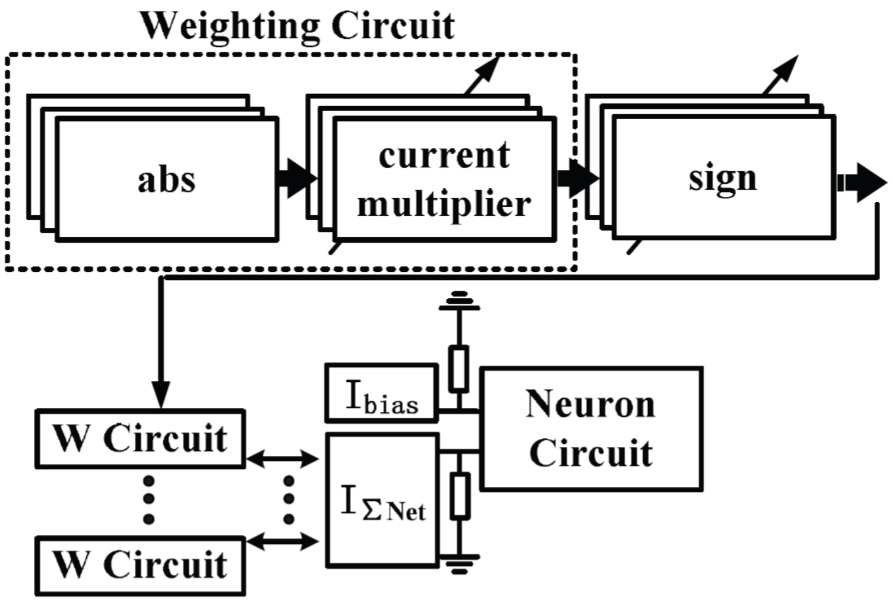


Figure 2.9: Bipolar Transistor Block Diagram Built in [9].

The next MLP design reviewed is proposed by Talaska *et al.* presenting a neural network that is not a MLP but is still useful in terms of power consumption and a figure of merit (FOM) [36]. The presented neural network is a winner-take-all (WTA) and is developed for distance calculation instead of classification. The WTA neuron presented has been developed to consume as low as 55 μW in one form of distance calculation. The authors also report a data rate up to 7 MHz. They use the data rate divided by the power consumption and multiplied by either the number of channels or the total number of neurons to determine the two FOMs they report. These FOMs are specifically developed for distance calculation circuits and not, however, for a general MLP structure.

The final two designs reviewed consist of neural networks that are not MLP-based but do offer highly comparative characteristics to what is proposed and achieved in this work. Park *et al.* demonstrate a deep learning network that is developed for mobile/portable devices while maintaining a high data throughput [37]. The authors implement a deep learning network that operates at a frequency of 200 MHz, consuming 213.1 mW at peak usage, and achieving a power efficiency of 1.93 TOPS/s/W. Tsai *et al.* detail a similar neural network with low power and high speed designs for both machine learning and Internet of Things applications [38]. The authors propose a network that achieves a maximum operation at 210

Table 2.1: Reviewed Neural Network Designs.

	Comp. Rate	Power	Syn. #	Rate per Syn.	Power per Syn.	FOM
Diotalevi *et al.*	30.00	3.00	1	30.00	3.00	-
Gatet *et al.*	2400	595000	10	240	59500	0.004
Bo *et al.*	1000	25000	28	35.70	892.90	0.040
Bo *et al.*	2500	200000	4810	0.52	41.60	0.012
Park *et al.*	411300	213100	2056	200	103.65	1.930
Tsai *et al.*	860160	310000	4096	210	75.68	1.450

MHz with 310 mW of power with only 41.3 pJ of energy being consumed per neuron weight in the system. These two works show recent state-of-the-art designs with a comparable FOM that is seen in this work's MLP system. Table 2.1 summarizes several of the neural network designs reviewed in this section. Going from left to right, the units for each column is as follows: computation rate in Mega-connections per second (MCPS), power in micro-Watts (μW), synapse number in integers, rate per synapse in MCPS, power per synapse in μW, and FOM in Tera-operations per second per Watt (TOPS/s/W).

2.2 Multiplier Review

One of the building blocks of the Multilayer Perceptron is a multiplier as seen in several of the figures in the previous section. Multipliers are constructed in a variety of forms depending upon the end-use applications and systems that integrate them. Generally, the multiplier in the MLP system functions as an operation to multiply the input signals, from either the intial inputs or the previous layer, with weight signals that determine the "importance" of a signal pathway based upon the overall MLP function [1]. These multiplication operations occur before the neuron in each layer, and each signal pathway has its own multiplier so that each pathway can be weighted individually thus providing the variances in the MLP system function.

By analyzing the analog MLP designs in the previous section, one can find that many of the designs use a Gilbert cell or translinear principles in the implementation of their multiplier circuit [4, 5, 33, 34, 7]. Gilbert developed the translinear principle decades ago and its use has spread throughout analog circuitry [39]. The basic translinear principle states that any closed loop containing an equal number of devices oriented in both directions (clockwise and counter-clockwise loops) creates a circuit where the product of currents in one orientation equals the product of currents in the other orientation [10]. The first Gilbert cells depicting this principle consist of bipolar transistors, whereas today the translinear principle has been expanded to include MOSFETs in weak inversion [40].

Figure 2.10 shows a simple translinear multiplier/divider bipolar circuit developed by Gilbert that uses bipolar devices. Equations 2.3 through 2.6 demonstrate the operation of this circuit and how it implements the multiplication and division.

$$J_1 J_2 = J_3 J_4 \tag{2.3}$$

Equation 2.3 is based off of the translinear principle equating two current loops of equal pn junctions where J represents the current density of a bipolar transistor.

$$J_4 = J_1 \frac{J_2}{J_3} \tag{2.4}$$

Equation 2.4 is a manipulation of 2.3 to isolate a single device's current density relative to that of the other three devices.

$$I_4 = I_1 \frac{I_2}{I_3} \tag{2.5}$$

$$A_1 A_2 = A_3 A_4 \tag{2.6}$$

Equation 2.5 represents removing the device sizing characteristics from the current and is only true when equation 2.6 is true. Equation 2.6 states that the device sizing (in this case the area of the bipolar devices) of the first two transistors must equal that of the second two in order for the current multiplication or division in 2.5 to be valid. Gilbert states in [10] that the devices need not all be the same size but rather only equal the same amount of area when their area is multiplied with the area of their corresponding device.

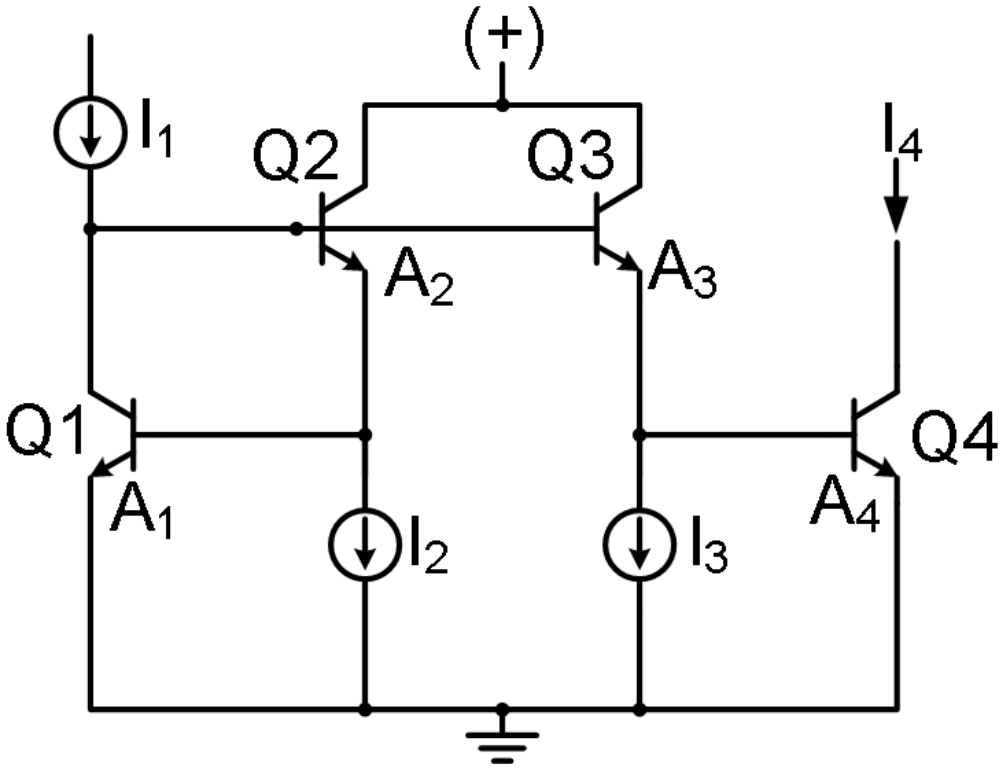


Figure 2.10: Transilinear Multiplier/Divider [10].

In [11], the Gilbert translinear principle is applied to obtain a multiplier that utilizes the back gates of two differential pairs to create a differential subthreshold output current for neural network applications. The authors proposed multiplier is shown in Figure 2.11 and uses only four transistors (along with additional bias circuitry) to implement both the multiply functionality as well as the sigmoid function for the differential current output. The downside of this topology is that the designer is required to have differential voltages as input signals while receiving a differential output current. Therefore, the overall system must then convert those current signals after a summing operation to voltages for the next layer in the neural network. These conversions will slow down the speed of the overall system even with the bonus of current signals being better suited to traveling along long traces to each neuron in a layer. Finally, the circuit can only receive one set of inputs from both the input paths and the weight paths which causes the overall system to be flooded with many copies of this circuit in order to perform at a typical neural network standard.

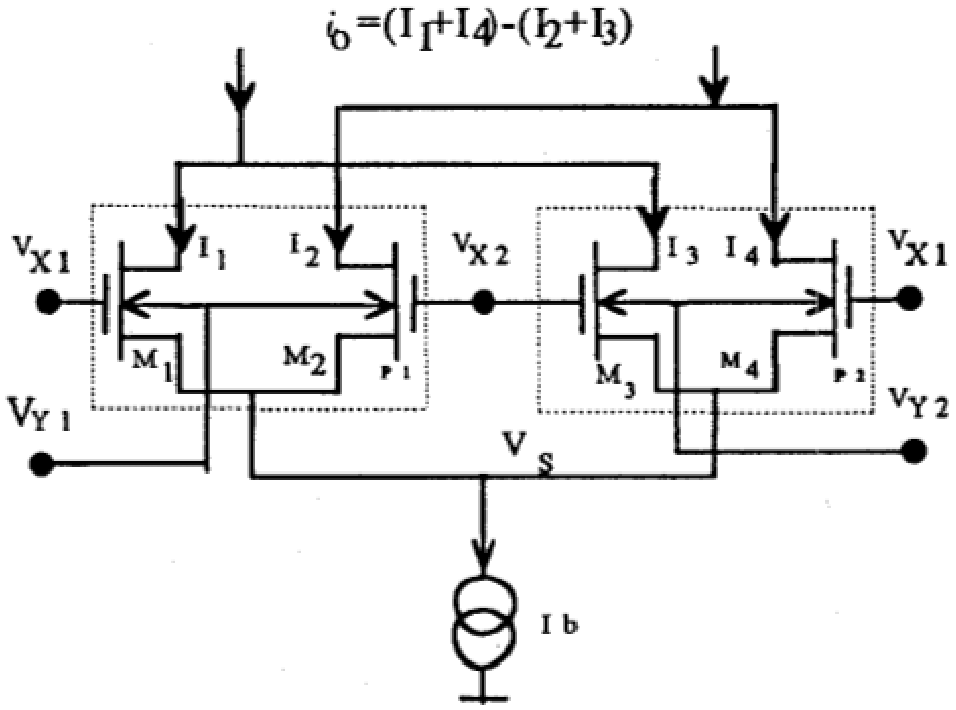


Figure 2.11: Proposed Four Quadrant Multiplier in [11].

The next multiplier design presented by Wilamowski details a circuit highly based upon Gilbert's original transilinear circuit but has been developed to accomodate all four quadrants in multiplication and division [12]. The circuit, seen in Figure 2.12, presents the basic translinear loop in MOSFETs M_1 through M_4 along with supplemental FETs that resemble a stacked cascode structure. The author states that by utlizing these transistors in the subthreshold region along with most of the FETs maintaing a zero gate-drain voltage the channel length modulation is considerably reduced as well as the threshold shift being similar for the important transilinear transistors. The major downside to this circuit structure is the amount of bias currents required to implement a single multiplication operation. Also, the circuit operation will be hampered as the voltage "headroom" (amount of voltage available to sustain a transistor in a particular operation region) for each transistor level is reduced due to smaller technology nodes requiring lower voltages. Even with the current mode operation, these restrictations limit the circuit too greatly to be used in a dynamic neural network system.

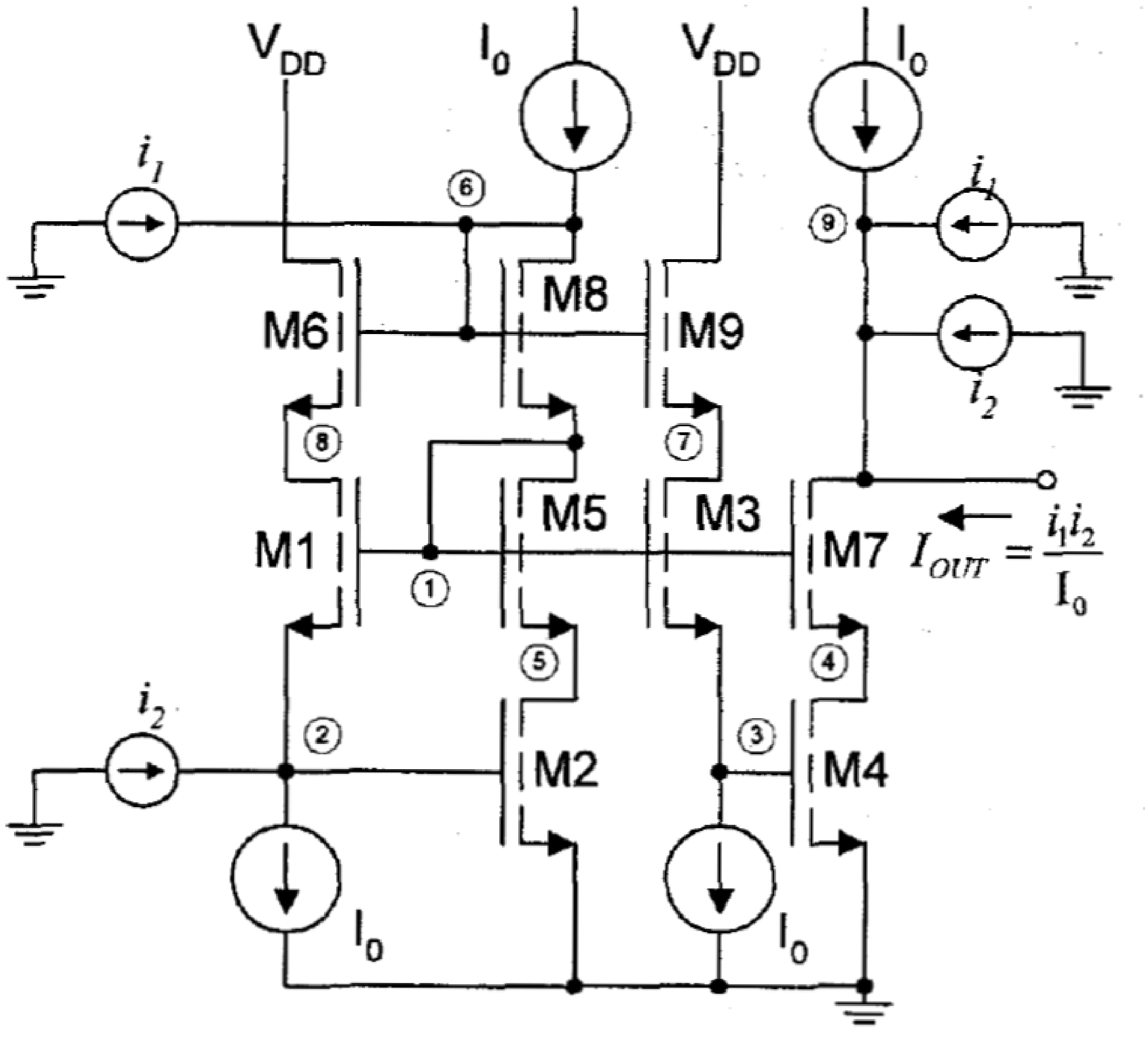


Figure 2.12: Proposed Four Quadrant Multiplier in [12] with Bias Currents.

Santos *et al.* propose a multiplier that utilizes the squaring properties of transistors operating in saturation alongside others in triode (linear or between the subthreshold and the strong inversion regions) [13]. Figure 2.13 details the two sections of the multiplier/divider circuit which consist of a geometric mean subcircuit and a squarer/divider subcircuit. The arithmetic of the overall multiplier/divider takes on the same form as the Gilbert multipler through the following manipulations:

$$I_{gm} = \sqrt{I_x I_y} \tag{2.7}$$

$$I_{out} = \frac{I_{gm}^2}{I_w} = \frac{I_x I_y}{I_w} \tag{2.8}$$

Equations 2.7 and 2.8 take I_x and I_y as the multiplier inputs and I_w as the weight input for the neural signal path. These subcircuits take on the same structure except for minor differences that transform the input and output of the geometric-mean subcircuit into its counterpart (input to output, output to input) as well as change the impedances from low to high (output to input transformation) or high to low (input to output transformation). As previously discussed, this circuit falls short in its voltage management for the "headroom" for each transistor in newer technology nodes. In addition, the multiplier/divider circuit must operate in the saturation and triode regions which will consume more power and voltage headroom to keep the transistors in the saturation region. The circuit also requires bias control voltages to be created elsewhere. These deficiencies limit its capabilities in a low power and high speed environment.

Baharmast *et al.* present another multiplier that utilizes the squaring characteristics of transistors [14]. Seen in Figure 2.14, this circuit provides a much simpler design with a lower transistor count and simplistic current mirroring to provide the multiplication operation. Much like the previous design, this circuit uses saturated transistors to produce the overall multiplication function described here:

$$I_{out} = I_{o1} + I_{o2} - (I_{o3} + I_{o4}) = \frac{I_x I_y}{8 I_B} \tag{2.9}$$

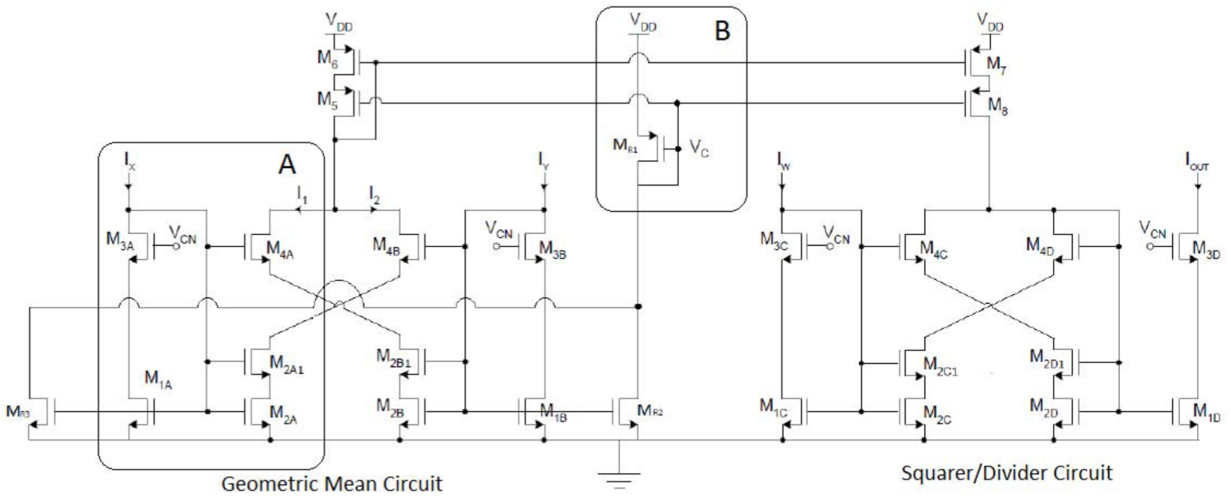


Figure 2.13: Multiplier/Divider Circuit Proposed in [13].

Equation 2.9 is an application of Kirchoff's current law at the output node and sums the desired currents from the four legs of the circuit with I_x and I_y being input currents. Though this circuit structure is simple in design, it will require additional support circuitry to provide copies of the input currents so that reliable copies can be summed together at the summed input as shown in Figure 2.14. Another shortcoming of this design is that there exists a limited input range for the currents such that they must obey this relation:

$$|I_{in}| \leq 4I_B \tag{2.10}$$

Where I_{in} is an input current and I_B is the bias current for the circuit. This input range limitation leads to a design that consumes more power and may not be easily translatable to subthreshold or lower power designs. Along with the input range limitation, this circuit design also has the potential for threshold voltage and transconductance paramter mismatches.

The authors in [15] propose a multiplier/divider circuit that utilizes the basic translinear loop methodology for a MOSFET. Figure 2.15 shows the multiplier design with differential inputs and outputs that use two single quadrant translinear loops. The simplified approach of this design will allow the circuit to fully utilize the current mode operation in terms of speed and power dissipation. The authors design this circuit to operate in the subthreshold region providing a low power capability for the multiplier. However, the major disadvantage of this

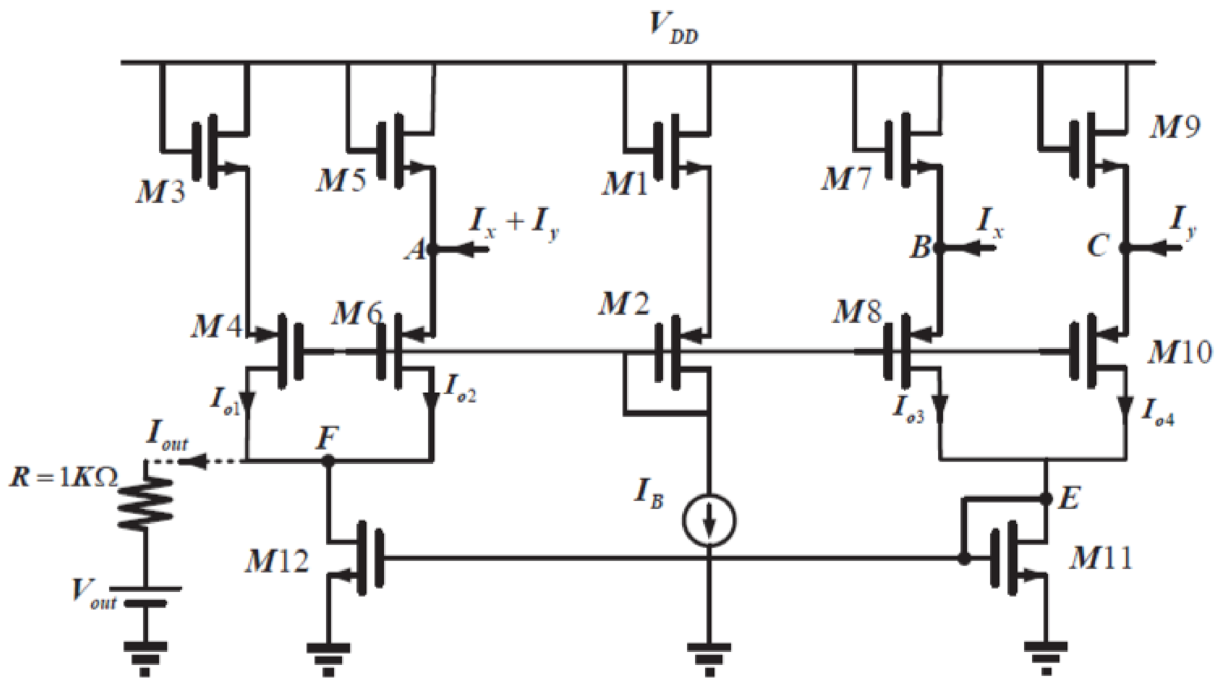


Figure 2.14: Multiplier Circuit Using Two-Quadrant Squaring Proposed in [14].

design is the need for differential current signals that will require specialized circuitry either on-chip at the beginning of the input signal pathway or off-chip in a commercial device. The differential signals are better at maintaining signal integrity but double the number of devices thus increasing the overall power for the multiplier. Along with the differential signals, the authors used relatively large transistor dimensions for different sections of the multiplier (a width of 20 μm for some transistors and a length of 10 μm for others). Lastly in the simulation results, the authors demonstrate that the design is only capable of operating effectively in the tens of kilo-Hertz frequency range rather than the faster speeds sought for this project (Mega-Hertz range).

The final multiplier design reviewed demonstrates another method for using differential and subthreshold currents to build a four-quadrant capable multiplier made up of two translinear loops [16]. The proposed circuit, seen in Figure 2.16, has the differential input I_G and I_W as well as a differential output in I_{syn}. The first tranlinear loop consists of transistors M5, M6, M9, and M10 with the second being composed of transistors M6, M7, M8, and M9. As discussed in the previous design, a pitfall of this circuit is its requirement

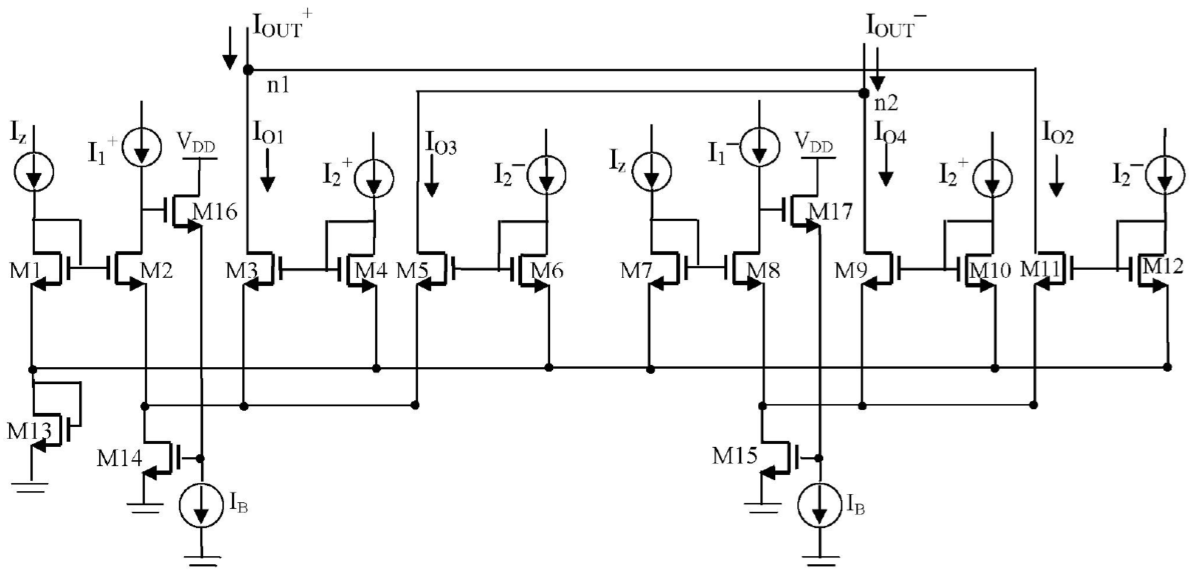


Figure 2.15: Multiplier/Divider Utilizing Translinear Loops from [15].

of differential currents (although these signals provide some cancellation of harmonics and reduced inteference and noise). Also as before, the authors chose large transistor dimensions (width equal to 40 μm and length equal to 18 μm for some transistors) in order to achieve a high linearity and $-3dB$ bandwidth of approximately 1 MHz. Though the authors provide good simulation results, they relate all the results back to ambiguous variables of x and w (for inputs and weights, respectively) that are not readily defined in the literature. Therefore, the more complex signal pathways and large transistor characteristics do not provide enough advantages to be utilized in a complex high speed and low power neural network.

In summary, the reviewed multiplier designs all have their own distinct advantages and disadvantages that must be analyzed properly in order to design a robust low power multiplier that is functional at a frequency in the ten's, if not hundred's, of Mega-Hertz range. Further multiplier design for this project will be discussed in more detail in the subsequent chapter.

2.3 Sigmoid Review

The second major building block of a Multilayer Perceptron is the activation function after the summation of the weighted signal pathways. This activation function is generally in the form of a sigmoid function (typically the logistic or hyperbolic-tangent functions).

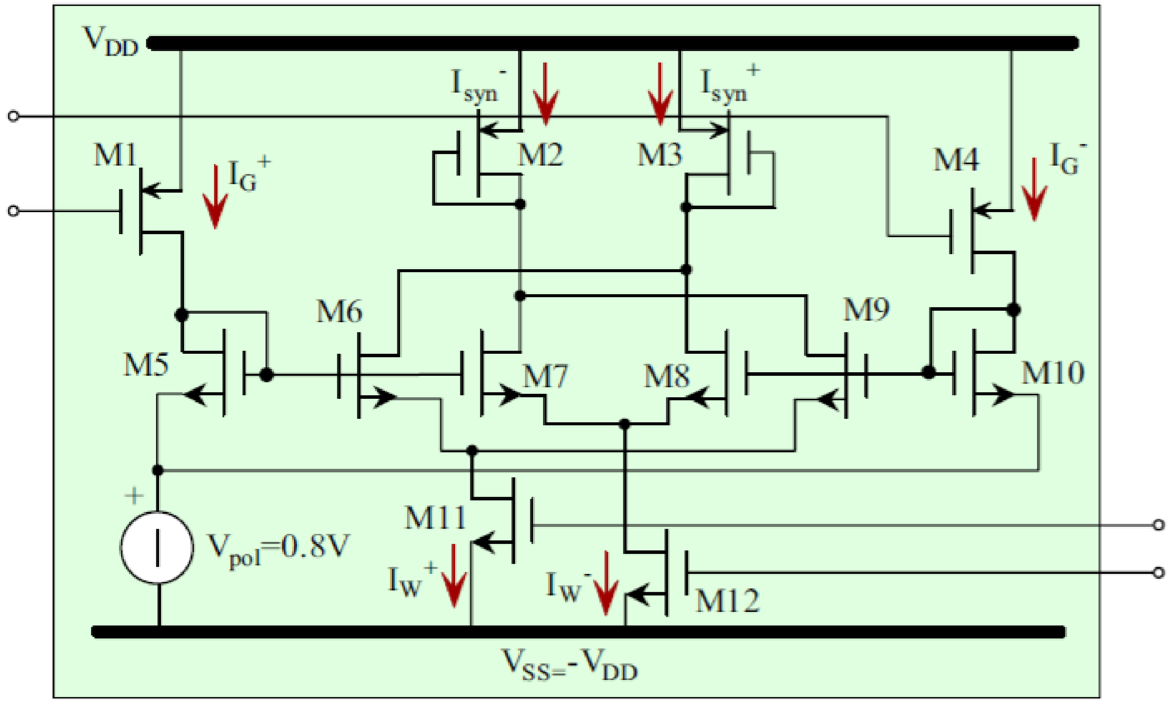


Figure 2.16: Four-Quadrant Multiplier with Differential Input and Output Currents [16].

Depending on the function used for activation, the function will categorize the received information either from 0 to 1 (logistic function) or -1 to 1 (hyperbolic-tangent function). This categorization will usually have a bias current or voltage that represents the analog range that the activation function will operate. For example, a bias current of 200 nA for a hyperbolic-tangent function will generally place the information in the range from -200 nA to 200 nA. The activation function will then either pass the information to the final output of the neural network or send it along another signal pathway to be weighted again for another layer's activation function. This project focuses on the logistic function as the chosen activation function for the MLP because of its need for only one positive power rail and no negative power rail as the hyperbolic tangent function requires.

The core circuitry for the hyperbolic-tangent (tanh) function consists of three transistors, a differential pair and a tail (bias) current [41]. The circuit is simply made up of three MOSFETs and is similar to Figure 2.8 in functionality. The differential output current of the MOSFET pair is quite similar to that of the Bipolar pair shown in equation 2.2 with some differences that stem from using FETs over Bipolar transistors. Analyzing the circuit from Mead, the following equations are generated:

$$I_{sat} = I_0 \exp(V_g \kappa - V_s) \tag{2.11}$$

Applying the saturated drain current to the differential pair yields:

$$I_1 = I_0 \exp(V_1 \kappa - V) \tag{2.12}$$

$$I_2 = I_0 \exp(V_2 \kappa - V) \tag{2.13}$$

The drain currents added together must equal the tail bias current:

$$I_b = I_1 + I_2 = I_0 \exp(-V)\left(\exp(V_1 \kappa) + \exp(V_2 \kappa)\right) \tag{2.14}$$

Solving for $\exp(-V)$ and substituting into equations 2.12 and 2.13 produces:

$$I_1 = I_b \frac{\exp(V_1 \kappa)}{\exp(V_1 \kappa) + \exp(V_2 \kappa)} \tag{2.15}$$

$$I_2 = I_b \frac{\exp(V_2 \kappa)}{\exp(V_2 \kappa) + \exp(V_1 \kappa)} \tag{2.16}$$

Taking the difference of equations 2.15 and 2.16 gives the final tanh function:

$$I_2 - I_1 = I_b \frac{\exp(V_1 \kappa) - \exp(V_2 \kappa)}{\exp(V_1 \kappa) + \exp(V_2 \kappa)} = I_b \tanh \frac{\kappa(V_1 - V_2)}{2} \tag{2.17}$$

For all of the above equations, κ is a constant that represents the rate of change of the MOSFET surface potential related to the rate of change of the gate voltage [41]. Again, the differences between equations 2.2 and 2.17 are minute and come down to the physics associated with the two different types of transistors. In either case, it is now well-established that a circuit including a differential pair will produce the desired sigmoid function.

The first of two sigmoid function designs as related to a neural network is presented by Maliuk and Makris in [6]. The hyperbolic-tangent circuit can be seen in Figure 2.17. As can be seen, the core of the neuron circuit consists of the aformentioned differential

MOSFET pair. In this design, the authors utilized current-storage-cells to provide bias and scaling currents for the differential pair and inputs, respectively. The circuit is completely current mode using several current mirrors to move the input signals to the differential pair. Transistors N5 and N6 are diode-connected (gate connected to drain of the transistor) under their respective current mirrors to provide a voltage at the source of N1 and N4 to be similar to the sources of N2 and N3, which are situated above the tail current transistor. This structure allows the subthreshold signals to propagate more efficiently through the current mirrors with as little mismatch as possible due to different drain-source voltages. The scaling current is used to control the slope of the hyperbolic-tangent function as it passes through its zero-point. The disadvantage of this design is the differential signals used and the requirements that are involved with producing and propagating those signals (pointed out in the previous section). Therefore, a single-ended version of this design could be potentially used for high speed and low power neural networks.

Valle and Diotalevi propose the second hyperbolic-tangent design that also includes programmable slope capabilities and subthreshold currents [17]. Figure 2.18 details the intricacies of the neuron block (typically referring to the activation circuitry) and how it connects to the overall neural network system. The programmable slope circuitry in this design, as compared to the previous one, is more complex and detailed in terms of how it interacts with the activation circuitry. As shown, the slope is controlled by the variable k that is an integer and scales the bias current for the programmable section of the design. The scaled bias current controls the tail current for a differential pair which will ultimately receive the input differential signals and then output differential currents to the activation circuitry. The hyperbolic-tangent activation circuit is similar to the previous design but better defines how the current will be output through diode-connected PMOS devices. This circuit design adds more complexity than may be necessary in order to add the slope programmability and also suffers from the differential signal issues previously mentioned.

In summary, sigmoid or activation functions can be easily created using a differential pair and tail current. The complexities come in how a design approaches different methods for input signal pathways and bias circuitry. Overall, it is feasible to have a subthreshold design that is capable of high throughput while still producing an adequate activation function.

2.4 Winner-Take-All Review

The Winner-Take-All (WTA) block is a circuit that is positioned right before the final output off of the chip and after the last neuron layer. The WTA circuit takes different neuron outputs and then compares them with each other making sure that the highest neuron output supercedes the others and is passed off-chip. For example, two neurons are outputting to the WTA circuit with current levels of 250 nA and 150 nA, respectively. The WTA block analyzes those two signals and determines that the 250 nA signal is the highest and passes it to the final output while suppressing the 150 nA signal. A WTA circuit can be designed to assess either voltages or currents and can output the highest value either in an analog or digital format depending upon the overall system design specifications for the integrated circuit.

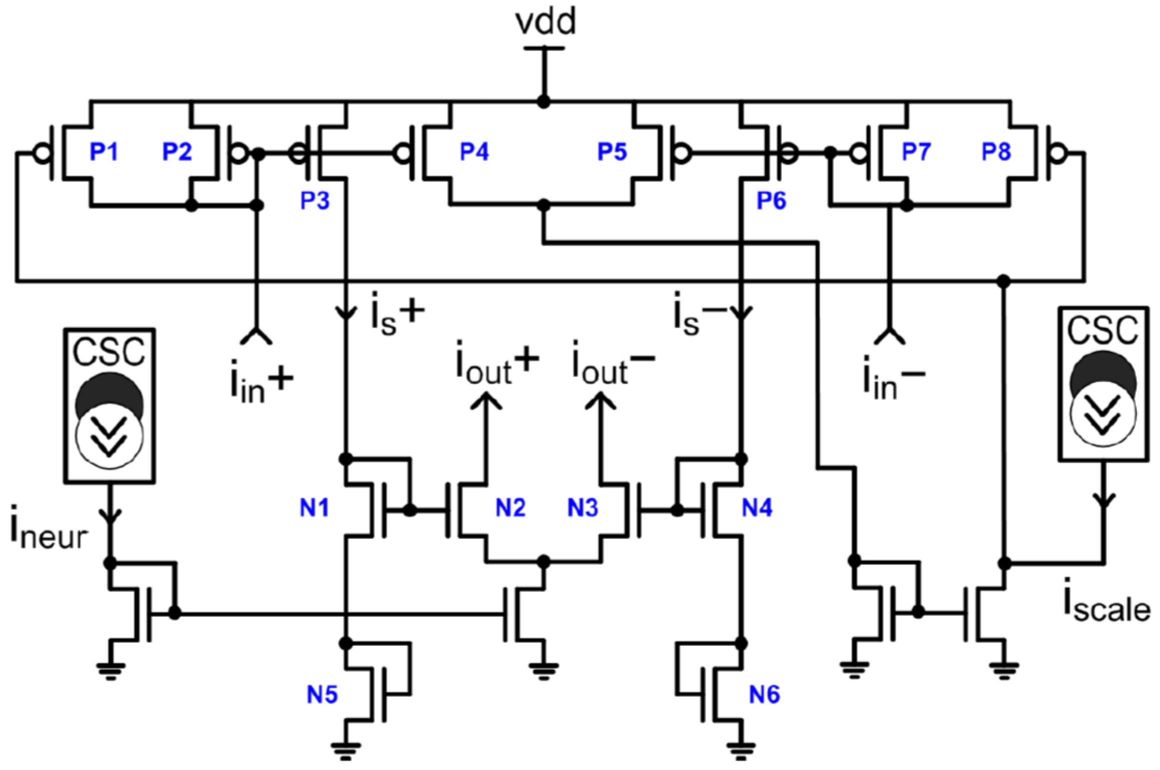


Figure 2.17: Neuron Consisting of Hyperbolic-Tangent Activation Function [6].

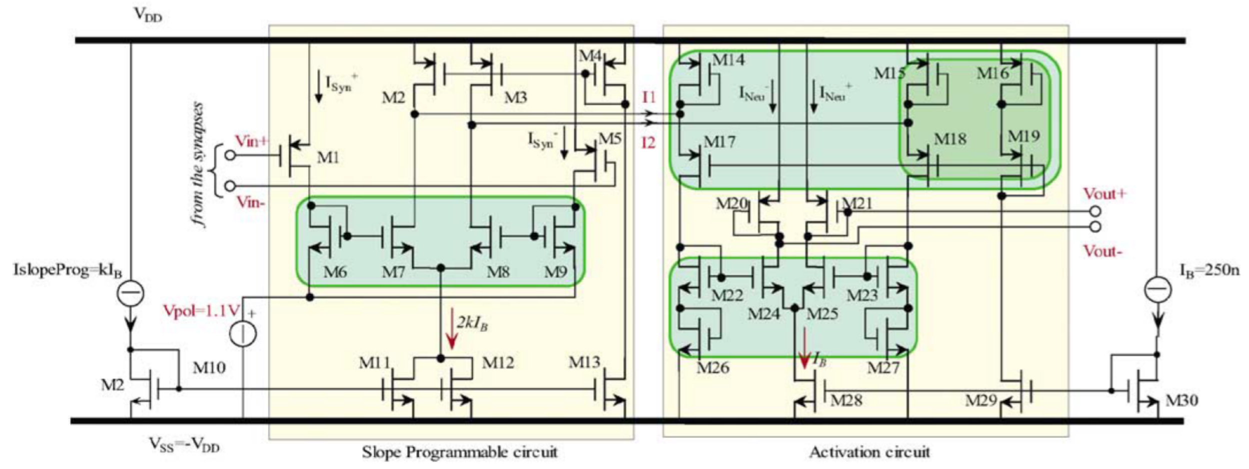


Figure 2.18: Neuron Block with Slope Programmable and Activation circuits [17].

The first WTA circuit reviewed is proposed by Lazzaro *et al.* and is a simple circuit that uses a common node to allow the winning voltage to resolve [18]. Figure 2.19 details a WTA circuit designed for two neuron current inputs. To add to this circuit, the design only needs to extend by two transistors for each consecutive neuron input. This WTA circuit is easily expandable and simple to implement in an overall system. The circuit is designed to operate continuously only changing as the inputs (I_1 and I_2 in this case) increase or decrease. Change in the inputs will alter the V_c node which is the winning response of the circuit. The disadvantages of this design include the circuit being unable to resolve high frequency signals and being unable to quickly resolve signals that are approximately the same level. Both issues could lead the circuit to produce a “false” output if structures off-chip are sampling the output periodically.

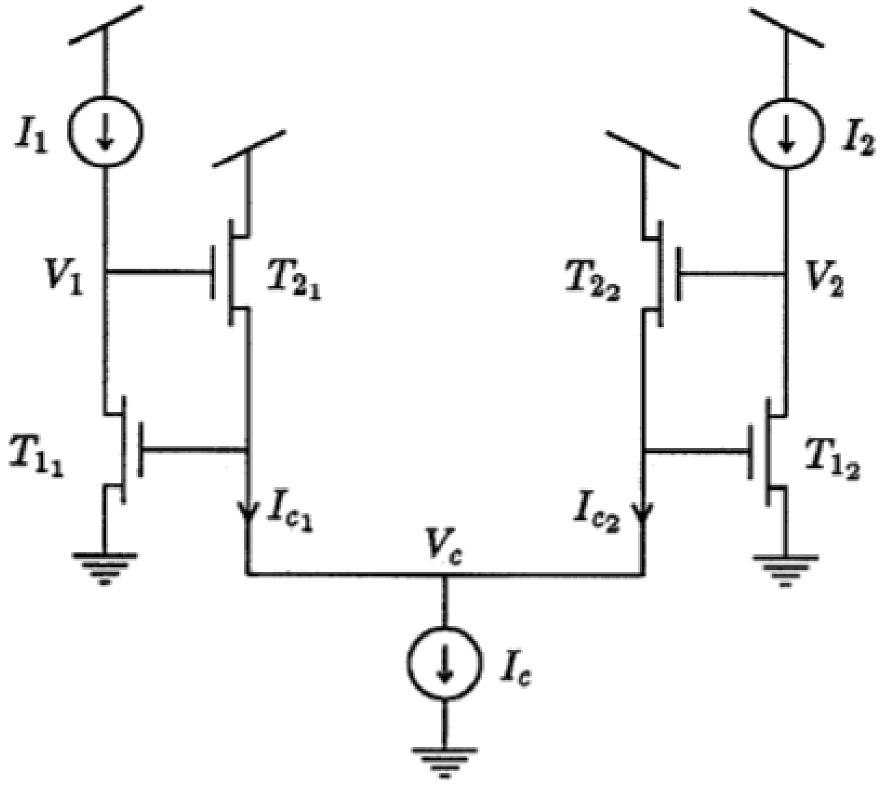


Figure 2.19: Two Neuron Winner-Take-All Circuit [18].

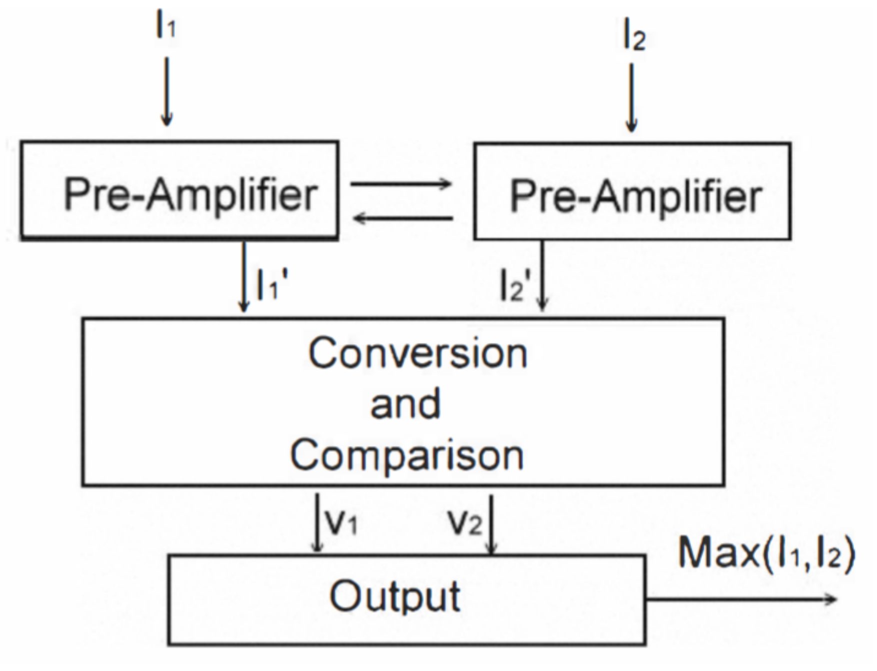


Figure 2.20: Winner-Take-All Block Diagram from [19].

The authors in [19] present a three-leveled WTA circuit that can be branched out to accomodate any number of inputs as desired. The WTA block diagram is shown in 2.20 that consists of two pre-amplifier circuits to boost the input currents, a current-to-voltage converter that then compares the two converted voltages, and an output block with two differential pairs that take the converted voltages and then outputs a current corresponding to the winning signal. This design is quite modular with the capability of stacking with other copies of itself to create a "tree" structure for 2^n inputs. The design is also capable of analyzing subthreshold currents that are less than the 100 nA threshold. The downside of this block structure is that it has added complexity from having three levels in a single WTA block. The input signals are required to move through these levels potentially slowing down the overall resolution time. The circuit also requires a synchronous reset signal in the conversion and comparison block which is additional support circuitry and signals that need to be generated for proper functionality of the WTA block.

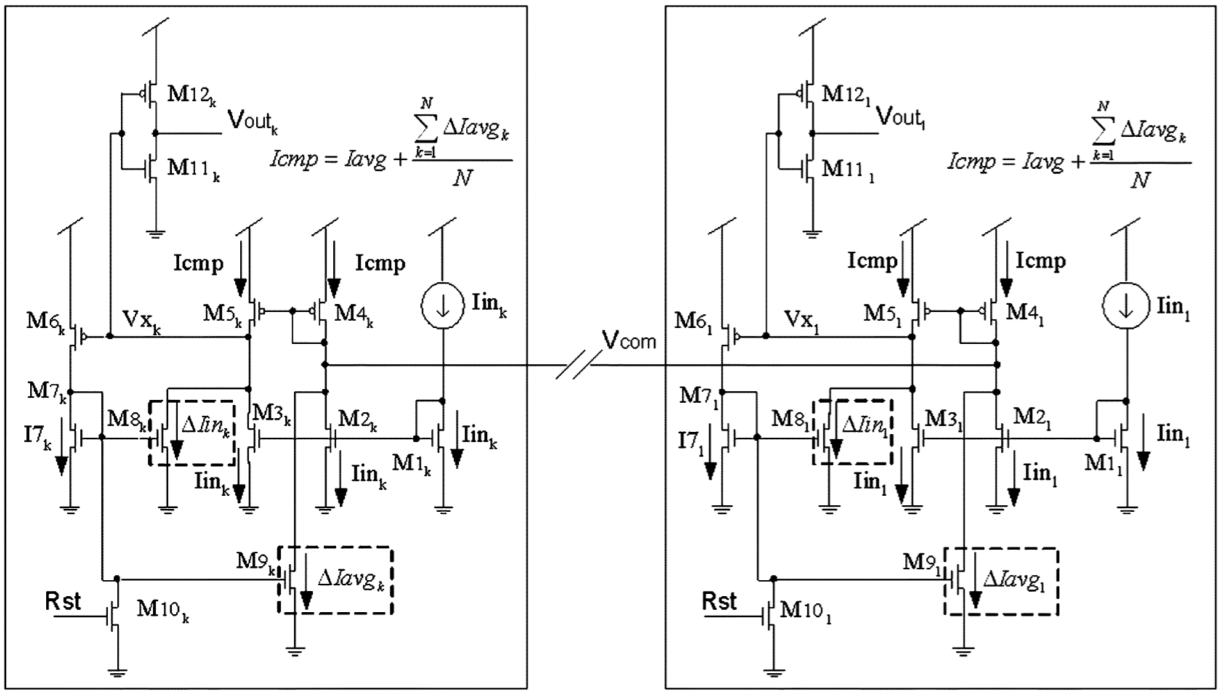


Figure 2.21: Winner-Take-All Circuit with Cells 1 and k [20].

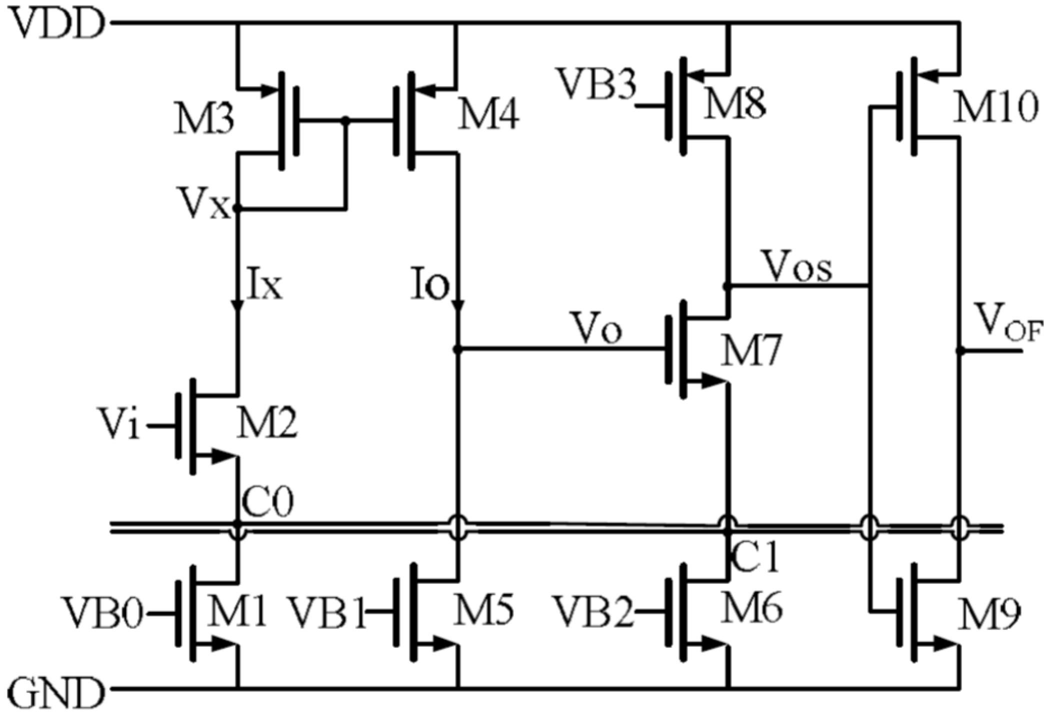


Figure 2.22: Voltage-In-Voltage-Out Winner-Take-All Circuit [21].

Fish *et al.* propose a WTA circuit that utilizes current inputs to produce voltage outputs and is highly expandable through the use of a common node [20]. Seen in Figure 2.21, the WTA circuit uses current mirrors to propagate the input current throught each cell as well as current comparison at the common node. As one cell receives a higher input current, the Vx_k node is pushed lower leading to a winning cell (if that cell has highest input current) and a high output voltage. The circuit makes use of two feedback components (excitatory and inhibitory through $M8_k$ and $M9_k$, respectively). These feedback paths are essential to the circuit resolving which cell wins the high output state. The benefits of the circuit include a digital output value, short resolution time (tens of nanosecond range), and high current precision (a few nano-Amps). The WTA circuit uses a reset signal much like the previous design, which is disadvantageous as described previously. The design also has a relatively high simulated power dissipation for subthreshold input currents (approximately 22.5 μW per cell). The resolution delay and power usage do not make this circuit ideal for high speed and low power neural network systems.

Padash *et al.* propose a WTA circuit that is a voltage-in-voltage-out configuration with a high frequency range of approximately 10 MHz [21]. The WTA circuit, shown in Figure 2.22, takes an analog input voltage and produces a digital output voltage. The resolution of the winning input voltage is settled through the two common nodes that connect all the WTA cells together. The circuit operates by converting an input voltage to a current that is then mirrored to produce another voltage, sent through a second comparator leg, and then the signal drives a digital inverter. The high speed and low number of transistors, as compared to some of the previous designs, make this WTA cell advantageous for higher speed neural networks. However, the WTA cell requires four separate bias voltages in order to properly function. Creating and maintaining these bias voltages requires several supplemental circuits increasing the system's overall power dissipation. Another disadvantage is the number of conversions from voltage to current and then back to voltage may increase the error in subthreshold neural network designs. If the accuracy of the WTA cell can be maintained at higher speeds as well as at lower power, then this design may be useful for high speed and low power neural network systems.

The authors of [22] have designed a WTA circuit that can be expanded through the use of four subcircuits that have been developed to determine the minimum and maximum of the desired number of input currents. Figures 2.23 and 2.24 detail the minimum and maximum subcircuits using PMOS and NMOS transistors and a two input WTA circuit utilizing the minimum and maximum subcircuits, respectively. This design boasts the ability to modularly expand to any number of desired inputs to produce a single output winning current. On the opposite side, the maximum PMOS and NMOS circuits make use of a latch that makes the resolution time (approximately 350 ns) for the output higher as the two transistors will fluctuate until the higher input current dominates. This WTA design requires its transistors to be in the strong inversion region thus consuming more power due to the higher currents. Therefore, this design may not be potentially useful for a low power, high speed neural network.

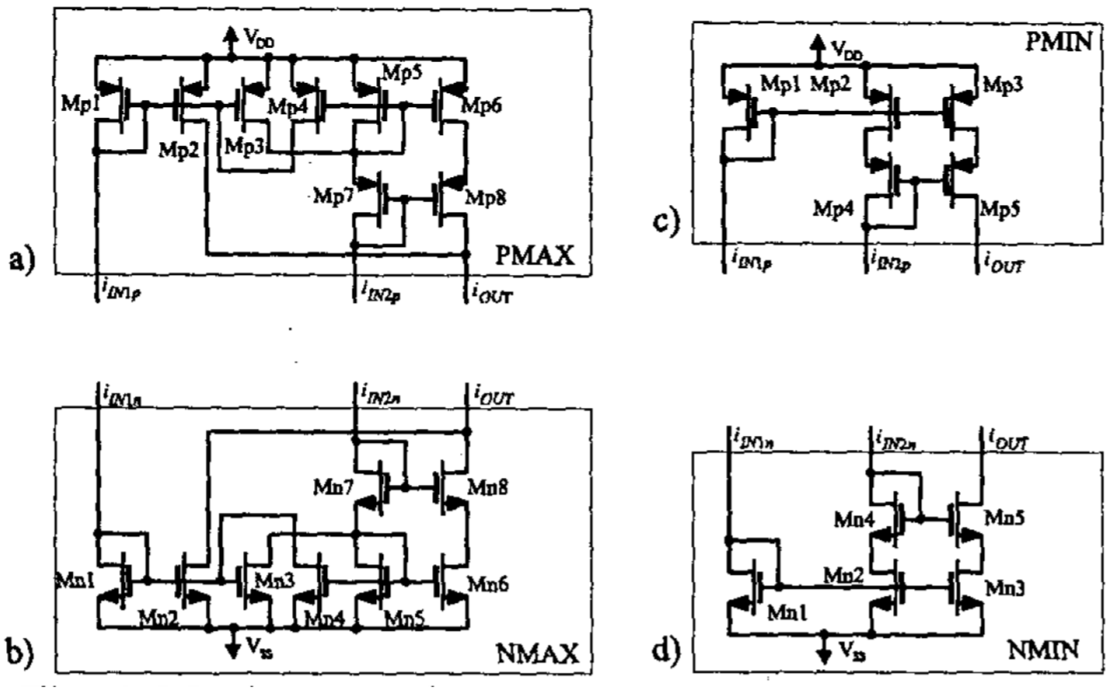


Figure 2.23: Minimum and Maximum Subcircuits for [22].

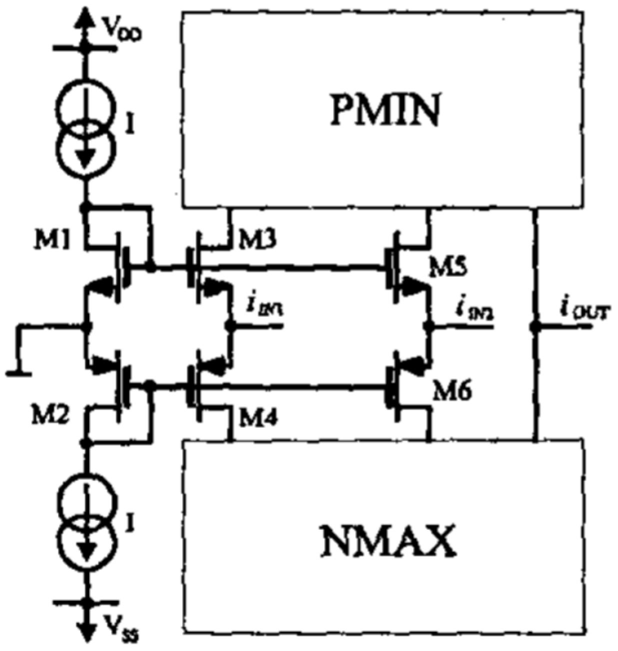


Figure 2.24: Two Input Winner-Take-All Cell Consisting of Min and Max Subcircuits [22].

Yu *et al.* propose a two stage WTA cell with a digital output in [23]. The core of the WTA cell consists of two stages that are identical to each other and are used to boost small differences in input voltages (less than 10 mV) high enough for the digital inverter to output the correct winning input signal. Each stage is made up of five transistors and can be seen in Figure 2.25. The input transistor converts the voltage signal to a current that is then amplified through the current mirror and sent on to the next stage for further amplification. The bias transistors (M_3, M_4, M_8, M_9) supply the necessary bias currents for the amplification. The circuit achieves a resolution time of 27 ns for small voltage differences and less than 20 ns for large voltage differences (greater than 50 mV). The downside is the circuit's specification of the transistors being held in the strong inversion region, which will lead to higher power dissipation. Though useful for small voltage differences, the use of two stages does not allow a resolution time necessary for higher frequencies (100's of Mega-Hertz). Therefore, this design is not suitable for a neural network with low power and high speed specifications.

In summary, the Winner-Take-All designs reviewed are the beginning point for a design that can be capable of low power and high speed functionality in a neural network setting.

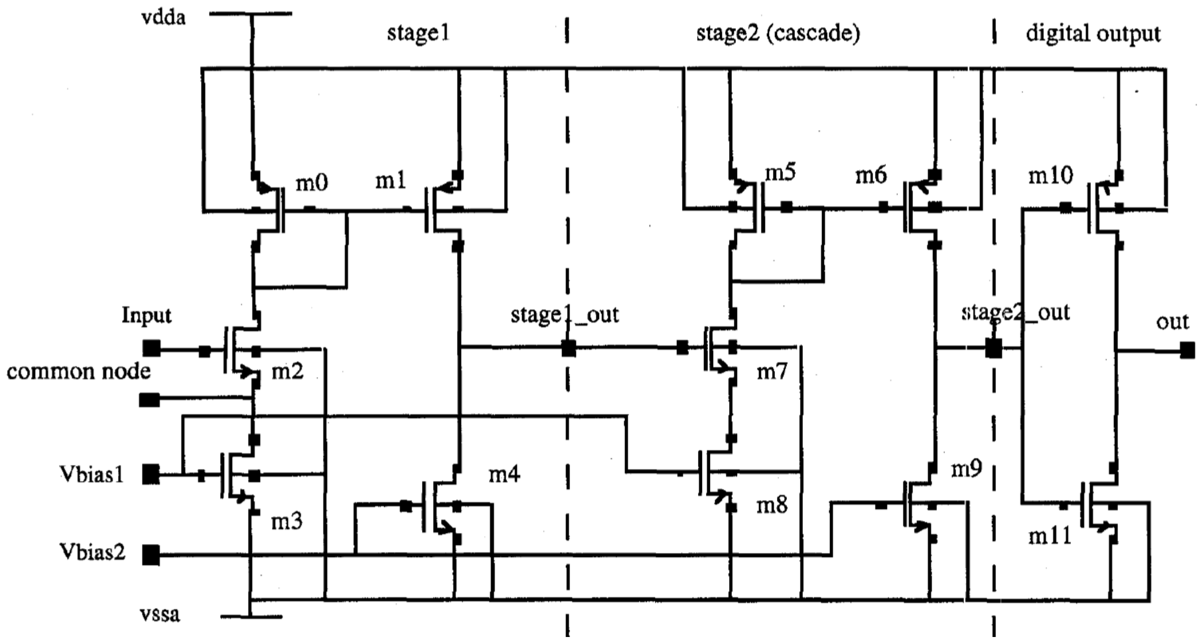


Figure 2.25: Cascade Winner-Take-All Circuit in [23].

Further analysis in some designs could push their potential to be viable for the needs of the proposed MLP.

2.5 Floating Gate Review

Floating Gate memory cells are an analog memory circuit that is composed of a voltage node that is “floating” (is not tied down by any interconnect or transistors). The floating gate is created by connecting the gates of transistors or MOSFET capacitors. This type of memory is categorized as a non-volatile memory in that it is capable of retaining its stored information for long periods of time without any power or signals being applied. The floating gate memory node is only subject to leakage currents through the gate oxide, which is minimal (typically on the order of femto- or pico-Amps). The cell is programmed using tunnelling and injection currents through the gate oxide to increase and decrease the floating gate voltage, respectively [24]. A floating gate structure was introduced previously in Figure 2.5 from [6].

Lu *et al.* present a deep machine learning system design that consumes a reported 11.4 μW of power after being trained and in recognition mode [24]. The reported work makes use of weak inversion circuits and current mode analog designs to aid in achieving this low power consumption. In addition to these design techniques, the authors utilized non-volatile memory in the form of floating gate transistors to store the necessary circuit parameters. The authors state that floating gate memory, when compared to a differential pair, provides several advantages including a similar transfer function, smaller occupied area, and the elimination of static bias currents. Figure 2.26 is the detailed schematic for the floating gate memory used in this system. The floating gate memory utilizes three voltage rails to incur either injection to add charge or tunneling to remove charge from the floating gate node. For both cases, the pulse width applied to the injection and tunneling voltage rails controls the movement of charge on the floating gate. Unlike the previous schematic in Figure 2.5, this design utilizes a PMOS capacitor to implement the charge tunneling. This work reports a peak energy efficiency of 1.04 TOPS/W (Tera-Operations per second per Watt) and is fabricated in a 130 nm CMOS process.

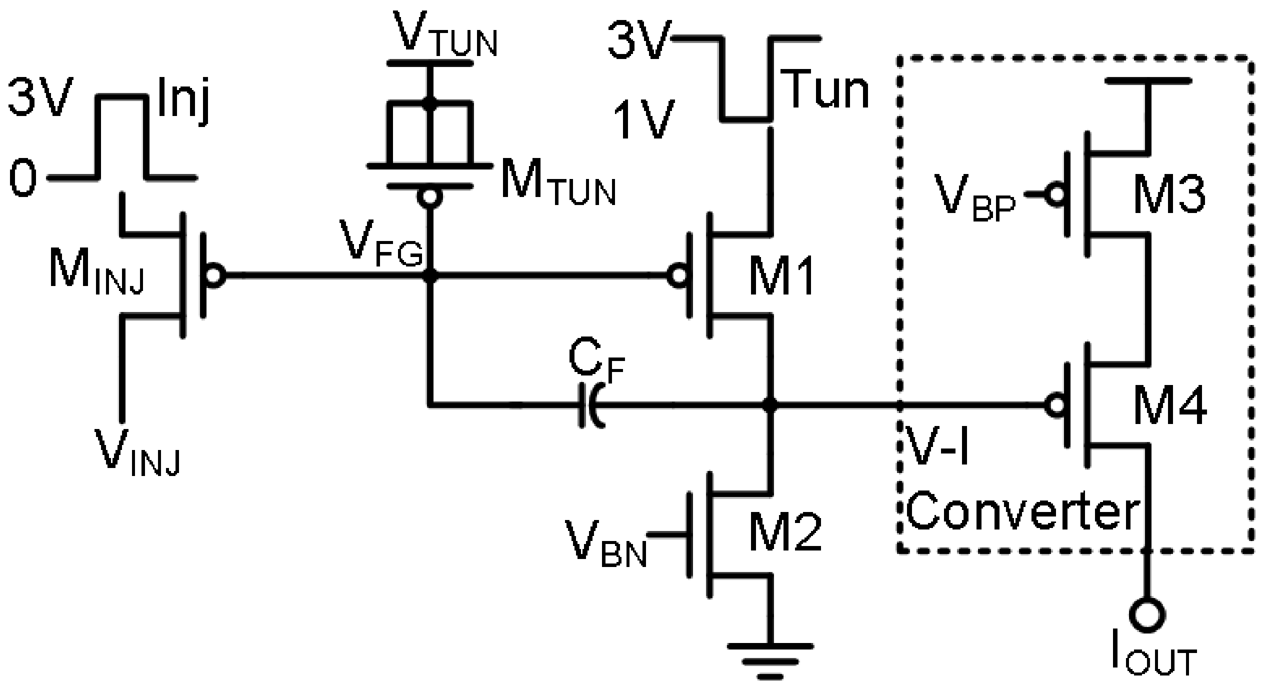


Figure 2.26: Floating Gate Transistor Memory Cell [24].

Wunderlich *et al.* present a field programmable mixed-signal array (FPMA) that utilizes floating gate switches and memory for configurability [25]. The authors use the floating gate structures to create several different types of interconnect nodes that can easily be programmed via removing or adding charge to the floating gate transistor with Fowler-Nordheim tunneling or hot channel electron injection, respectively. Through the floating gate interconnect switches, the authors create a routing network that can easily direct a signal down the proper pathway for the configured circuit in their FPMA. The FPMA is implemented in a CMOS 0.35 μm process. The floating gate transistors are stated to have a programmed voltage that has a higher dynamic range which leads to increased performance in speed, power, and signal integrity while having reduced density when compared to other conventional storage and switching devices. Figure 2.27 depicts the different types of floating gate switches utilized in this work. Figure 2.27a is the standard floating gate cell whereas Figure 2.27b is a floating gate cell setting the input voltage of an inverter. Figure 2.27c and 2.27d represent two abutting or two crossing signal lines being connected via a floating gate switch, respectively. Figure 2.27e is a s-switch implemented with six floating gate transistors. The s-switch allows an entering signal to be routed in a variety of useful manners, adding

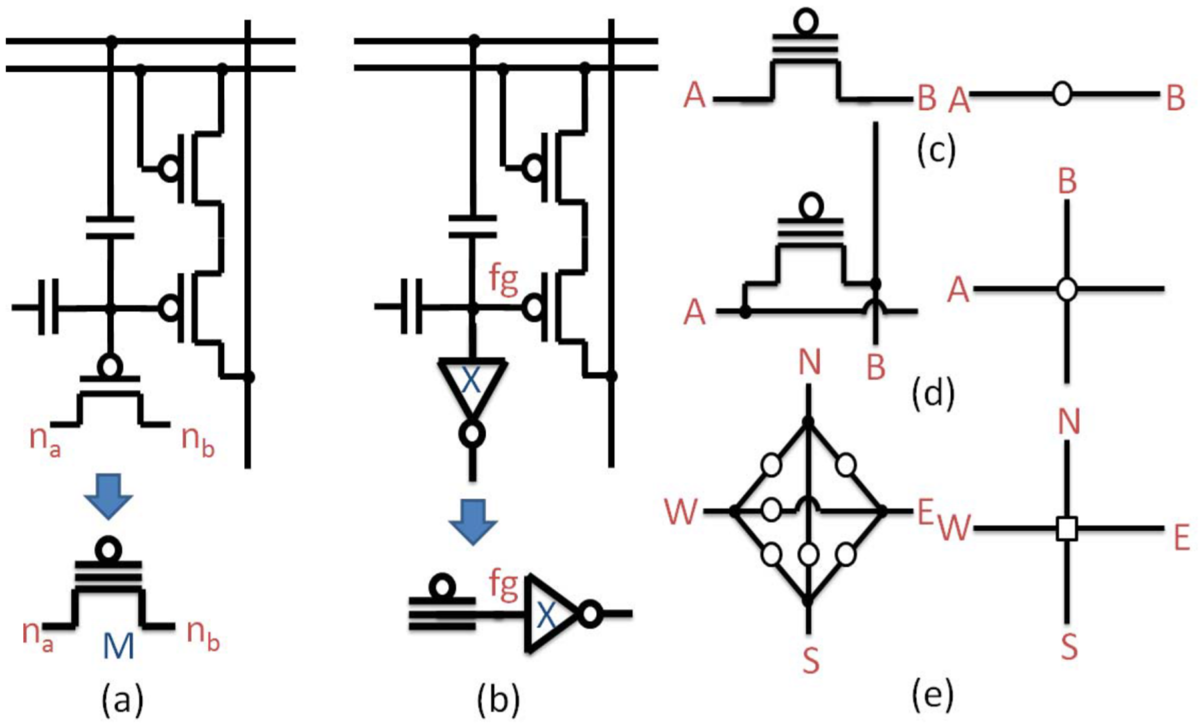


Figure 2.27: Floating Gate Switch Cells [25].

increased configurability. Therefore, the floating gate switches can be used to create a programmable structure in a neural network system.

In summary, floating gate memory cells can be used in several fashions as either a voltage storage cell, a voltage cell that converts to an output current, or a voltage cell that controls a transistor acting as a switch. Also, floating gate cells are power efficient as they inherently do not suffer from large leakage currents. The use of these cells is highly desirable for a neural network system that has design specifications to be configurable and low power if the complexity of the system does not hamper the programmability of the floating gates.

2.6 Summary

The reviewed literature provides a reasonable starting basis for designing the MLP system in this work. With respect to other MLP and neural network designs, they do not achieve both a high speed/low power design and a configurable architecture. The designs range greatly in the number of neurons or synapses in their respective neural network that in turn increases the power usage seen by each design. Decreasing the power per neuron or synapse in addition to adding configurability will greatly improve the overall system functionality. The reviewed multiplier circuits detail that simplier circuits that are similar to that of Gilbert's cell (seen in Figure 2.10) will function appropriately at higher data rates as well as consume less current (as there are less transistors for leakage paths), especially in the subthreshold region of operation. The sigmoid circuit review takes a similar approach to that of the multiplier in that subthreshold operation is beneficial to achieving the desired sigmoid function out of MOSFET transistors as that of BJTs in a differential pair design. The WTA designs provide a good basis for building a circuit that can operate with currents at high speeds with minimal space utilized while providing accurate outputs. However, a current comparison (similar to Figure 2.22) could be highly advantageous to create a thresholding circuit for inverters to digitize. Lastly, floating gate switches are highly advantageous as nonvolatile memory/switch choices but are also very complex to program if hundreds are desired in a system. Therefore, simple single transistor switches are more desirable than their floating gate counterparts as they offer simpler programming and less area usage.

Chapter 3

MLP System Design and Implementation

3.1 Multiplier Design

The multiplier is an important base element to a neuron in the MLP. The multiplier allows weights to be integrated with the input signal propagating through the neural network. After looking at the multiplier designs in Chapter 2, the past designs are used to provide a starting design point for the multiplier used in this MLP neural network. In addition to the older multiplier designs, the utilization of current signals suggests another design criteria for the initial multiplier design to meet. Therefore, a starting design similar to Figure 2.10 is used in the development of the multiplier for this system. It should be noted that there are no width and length dimensions explicitly stated on any of the design figures that follow as the minimum width (360 nanometer [nm]) and length (240 nm) are used for all transistors in the system.

Figure 3.1 is the final design obtained to accomodate both current signals and voltage headroom for the signal pathway to effectively propagate the desired input/current signals. The core structure of this multiplier design (left circuit in Fig. 3.1) almost mimics the Gilbert cell shown in Figure 2.10. The five core transistors operate similarly to the Gilbert cell because of the MOSFET's operation in the subthreshold region allowing an exponential function in the drain current similar to Bipolar transistors. This similar functionality creates

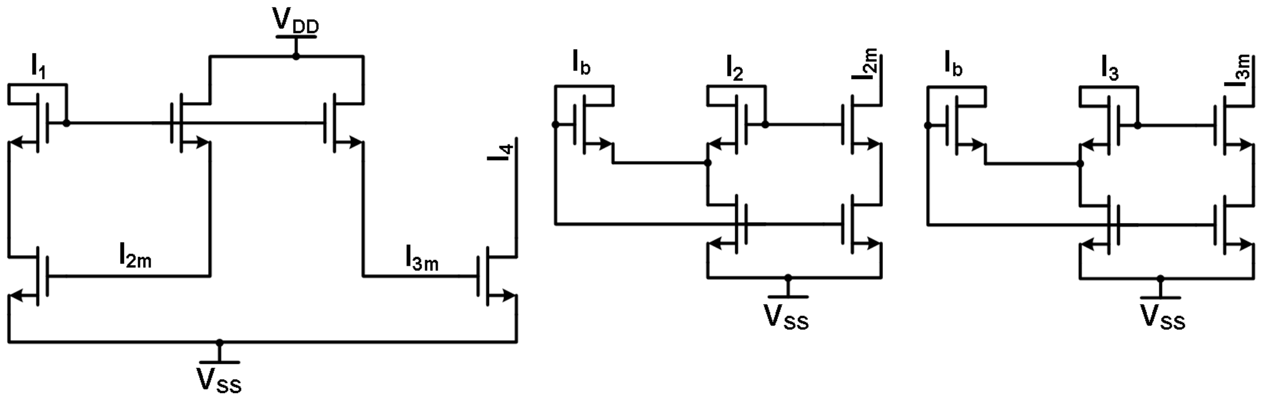


Figure 3.1: Multiplier Circuit Schematic.

the same output function seen in Equation 2.5. The addition of the cascode device in the I_1 input allows the multiplier to operate faster while maintaining signal integrity as well as a high impedance input for the current signals.

The rest of the circuits in Figure 3.1 represents two Minch cascode biasing schemes that allow the weights to adjust the input signal. The Minch cascode circuit from [42] creates a low voltage cascode structure that allows for weights in the subthreshold region. The additional benefit from utilizing the Minch cascode is the ability for the circuit to operate effectively with a low voltage ceiling (i.e., requires little voltage headroom). The Minch cascode circuit also provides a more accurate current mirroring operation than other simplistic current mirror designs. This effect allows for the multiplier weights to be more reliably reproduced as they are programmed even with transistor mismatch and operation in the subthreshold region. The key to the accuracy of the Minch cascode comes from the input bias current signal (I_b) that helps stabilize the current mirror operation with the additional bias current. In addition to a better mirrored current, the Minch cascode scheme allows the transistors to operate with minimal voltage headroom allowing for voltages to swing effectively if necessary. Since the weight signals are DC values and do not require high speed operation like the input signal I_1, the Minch cascode can easily be integrated into the Gilbert multiplier while maintaining signal integrity.

3.2 Sigmoid Design

The sigmoid circuit is the other important base element to the neuron in the MLP system. The sigmoid creates a logistic function or hyperbolic tangent function output (Figure 3.2) at the end of the neuron signal pathway that will either propagate onto another neuron or to the winner-take-all block. For the MLP system, the logistic function is utilized since it does not require a negatively biased circuit to operate as in the hyperbolic tangent function. The sigmoid designs in Chapter 2 are used as a starting point for the sigmoid circuit used in the MLP's neurons. The utilization of current signals in the sigmoid circuit is again desired and designed for during the development of the final circuit topology. The starting sigmoid circuit design most closely resembles the design in Figure 2.17.

Figure 3.3 is the final design obtained that accomodates high frequency current signals as well as outputs the desired logistic function. The sigmoid design is composed of several current mirrors that relay the input, bias, and output signals to and from the differential transistor pair at the core of the circuit. The basic functionality of the circuit is that when the main input signal I_p is below the reference input signal I_n, the output signal I_{out} will be near zero current. When I_p becomes much higher in magnitude than I_n, I_{out} will then output a current that is near that of the bias current I_b for the sigmoid. This behavior is typical of a logistic function at the positive and negative extremes on the horizontal axis. The reference current I_n allows for the logistic function's half point to be shifted further up the horizontal axis. This functionality requires the circuit to have more input current in order to reach the fully saturated bias current output. Performing a DC sweep on the input

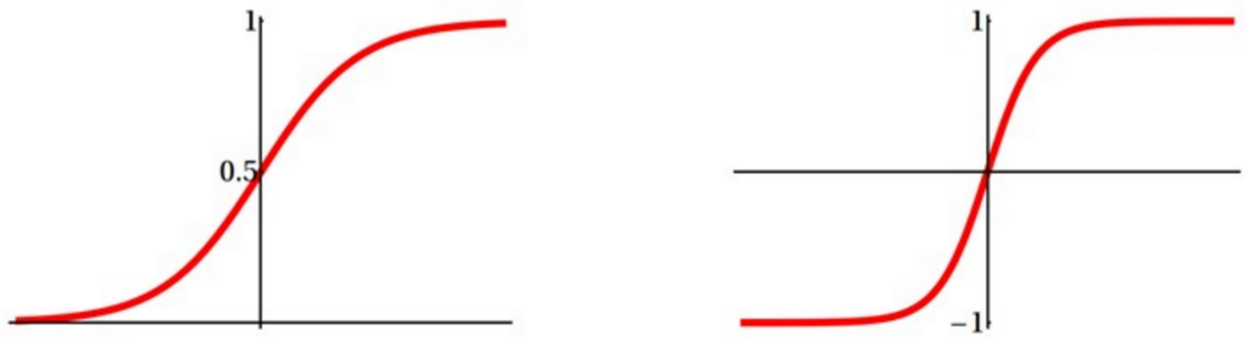


Figure 3.2: Logistic Function (left) and Hyperbolic Tangent Function (right).

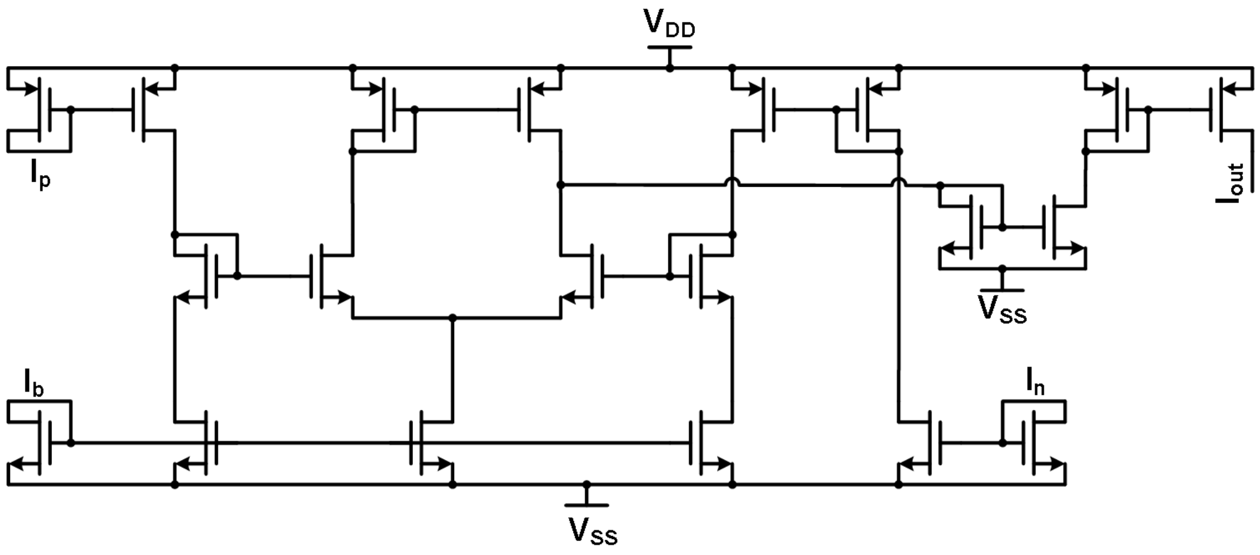


Figure 3.3: Sigmoid Circuit Schematic.

signal I_p would create an output signal similar to that of the logistic function seen in Figure 3.2.

For this sigmod design, all of the input and output signals are designed such that the circuit has a structure that interfaces easily with the multipliers before and after it. This fitted structure requires that the input signal I_p and the output signal I_{out} are PMOS current mirrors to source current into the NMOS devices at the input and output of the multiplier in Figure 3.1. The reference current I_n utilizes a NMOS current mirror input as it receives its input signal from DC bias circuits much like the bias current signal I_b. The bias current I_b sinks current for the two input signals as well as the differential transistor pair to ensure proper biasing throughout the core sigmoid circuitry and that the output signal has a maximum value of the bias current. The functionality of the sigmoid circuit can only be obtained by operating in the subthreshold region, which enables the output to take the shape of the desired logistic function for proper signal propagation.

3.3 Thresholding Circuit Design

The final standalone circuit before the MLP system circuits is that of the thresholding circuit (TC) that is based off of the winner-take-all (WTA) circuitry in Chapter 2. The combination of the TC and a 2-input OR gate produces a similar functionality to that of the WTA circuitry. The TC receives neuron outputs and outputs a signal based upon the highest signal level it receives at its input. The WTA circuits in Chapter 2 provide a great deal of background information and provide designs to start the development process for a high speed current capable TC. Figure 2.22 provides a good initial design of a current capable TC circuit. A similar structure of two signals "fighting" against each other to create an output signal based off of a comparison is desired for the MLP system outputs.

Figure 3.4 shows the block diagram for the two WTA structures that correspond to the two MLP system outputs. Each "complete" WTA design consists of a multiplier circuit (Figure 3.1) and a TC cell (Figure 3.5). The multiplier circuit sums the neuron current outputs at its input terminal and then scales the summed currents in order to create a better signal for comparison in the TC cell. The TC cell takes the multiplier output and compares it to a reference current level. This comparison determines if the signal should remain "high" or "low" by creating a voltage at the comparison node that is either just

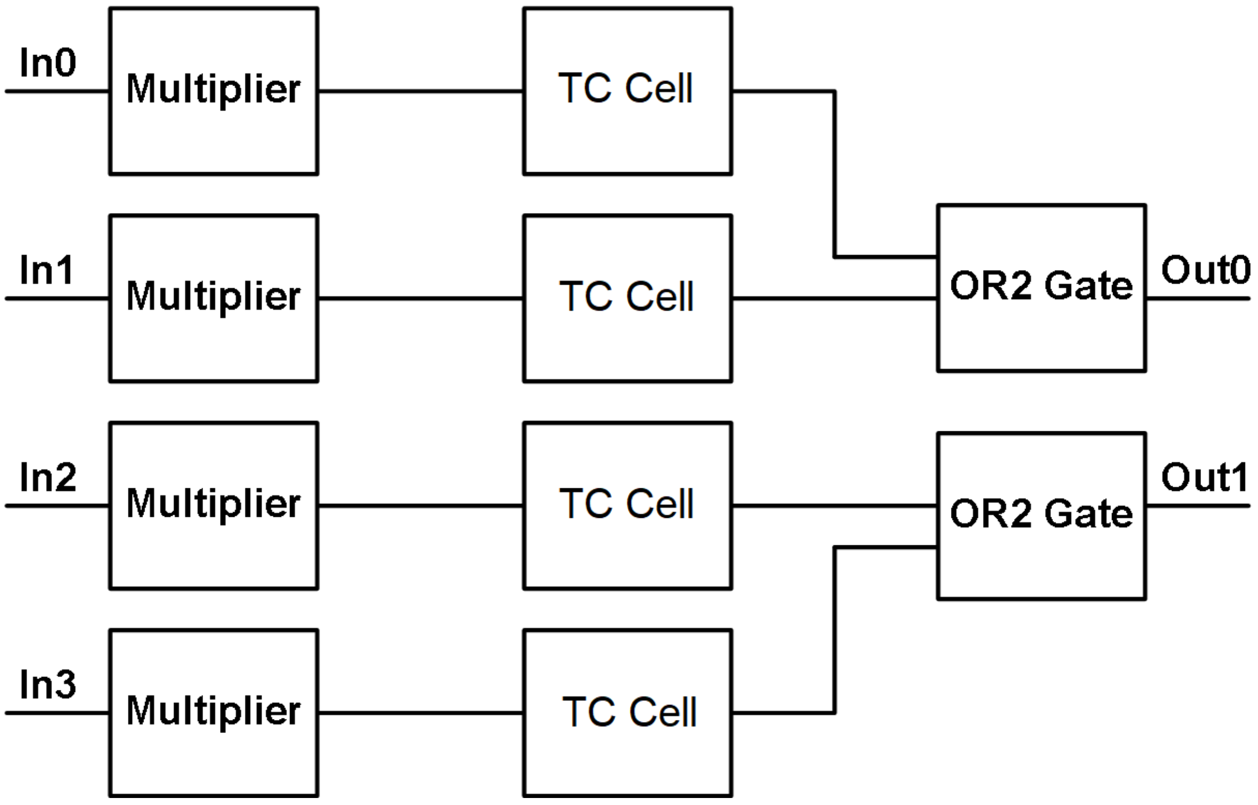


Figure 3.4: Winner-Take-All Block Diagram.

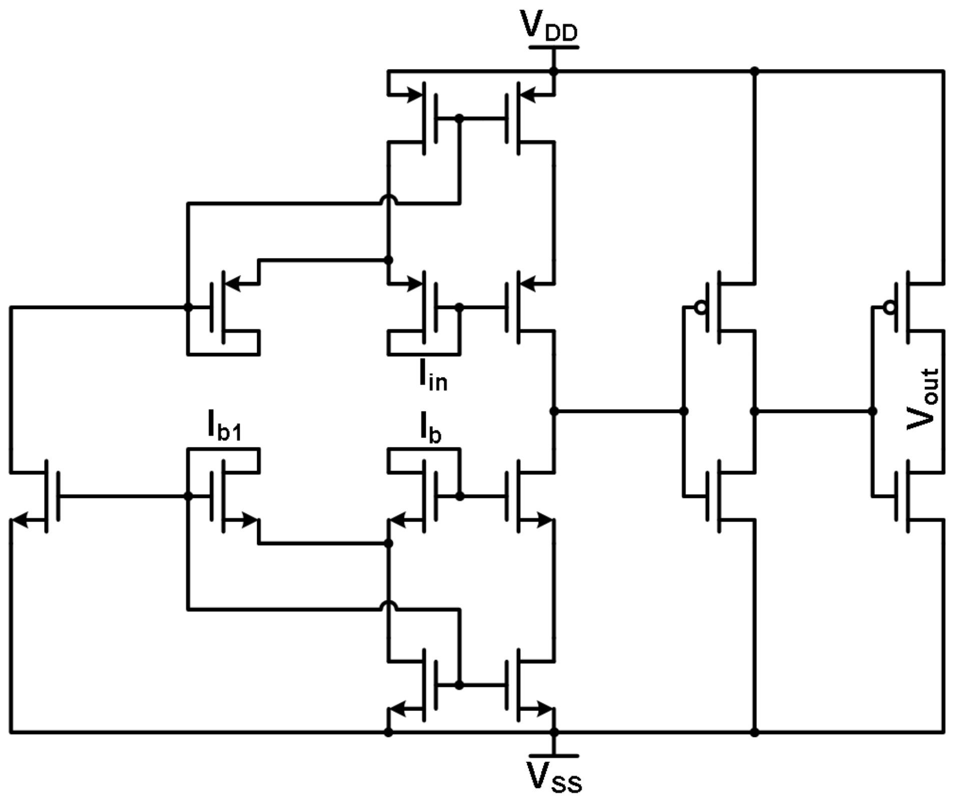


Figure 3.5: Thresholding Circuit Cell Schematic.

above the threshold voltages of the following inverters or just below their threshold voltages. The use of the inverters allows the final output off-chip to be a digital voltage instead of an analog current signal that then needs to be converted. The inverters have different minimum widths (160 nm) and lengths (120 nm) than that of all the other transistors to provide faster functionality as they create the digital output voltage as well as utilize less physical space.

The TC cell in Figure 3.5 contains three current inputs and one voltage output. The main input signal I_{in} that comes from the multiplier (and the neurons before that) goes into a PMOS Minch cascode current mirror in order to maintain signal levels and integrity before the comparison node. The reference current I_b for current comparison is input into a NMOS Minch cascode current mirror similar to those in the multiplier circuit mentioned previously in this chapter. The Minch cascode structure requires a bias current I_{b1} in order to operate effectively. The reference and bias current inputs are taken from system circuits and are DC values. The inputs I_{in} and I_b are mirrored and compared against each other at the comparison node before the two inverters. As mentioned before, this node fluctuates based upon the input signal I_{in} around the inverter threshold voltages. The output signal V_{out} is a digital voltage signal that is then passed on to an OR2 gate. The OR2 gate provides another level of comparison with another signal chain ensuring that the MLP system output follows the Winner-Take-All concept.

3.4 MLP System Design

The overall MLP system consists of the three aforementioned circuit blocks as well as several support circuits. The support circuitry consists of numerous copies of biasing cells, switching cells, and shift registers that provide power, connectivity, and configurability, respectively. The whole system contains two separate MLP structures. The smaller, simpler MLP structure is used for initial testing and basic programming tests to confirm functionality of the chip and is shown in Figure 3.6. The larger, more complex MLP structure is the main neural network design intended for low power and configurable use. The block diagram for the main MLP is detailed in Figure 3.7 minus the bias and shift register support circuitry. Both MLPs are designed to operate independent of each other and with the same capabilities.

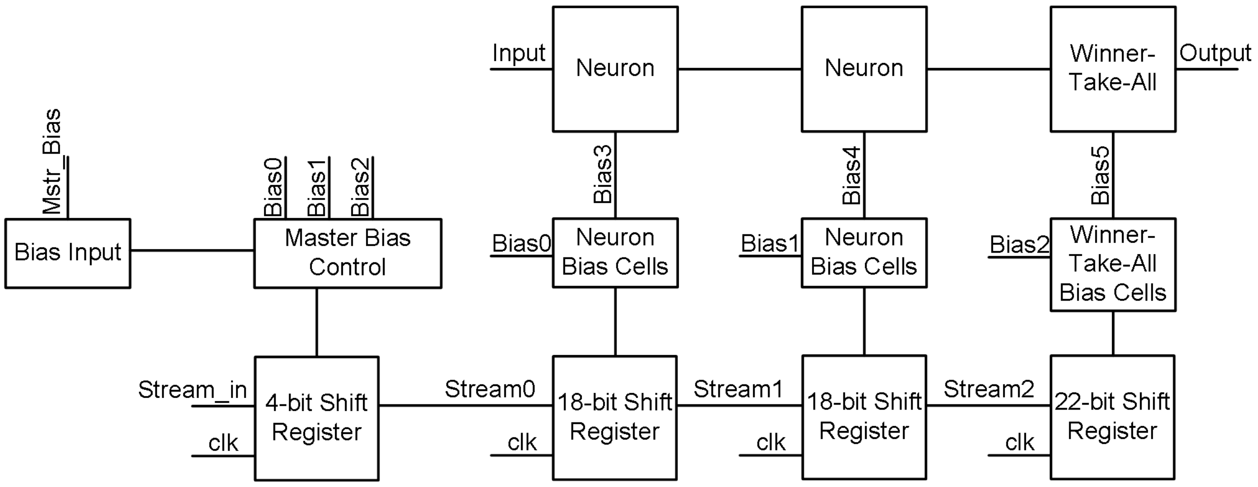


Figure 3.6: Simple MLP Test Circuitry for Chip Functionality.

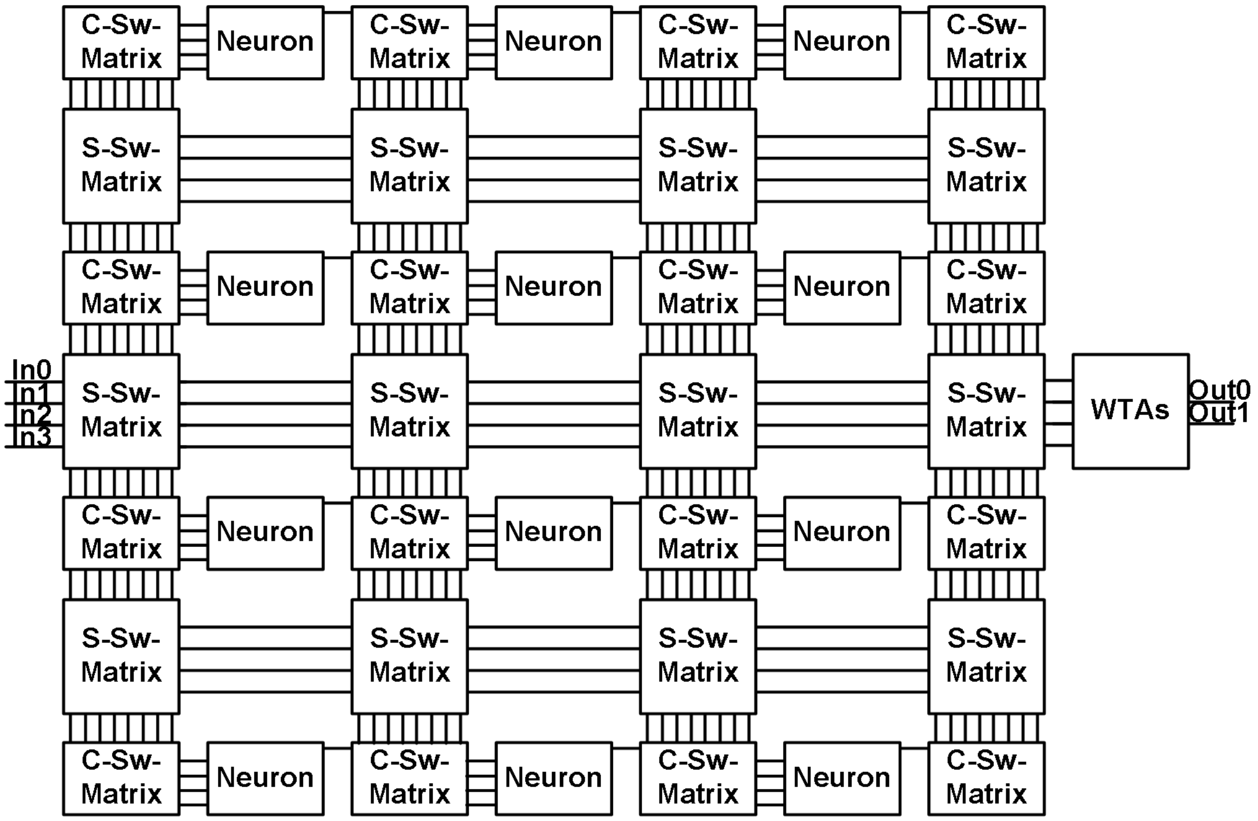


Figure 3.7: MLP Block Diagram for Configurable Low Power System.

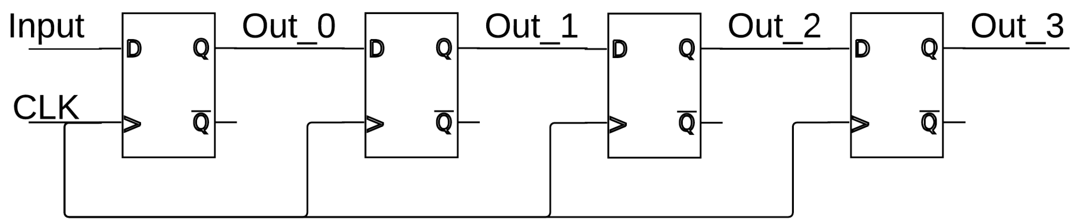


Figure 3.8: 4-bit Shift Register made up of D-Type Flip-Flops.

The simple MLP architecture in Figure 3.6 is a more managable initial testing structure as the programming to initialize the biasing for the neurons and winner-take-all block is much less complex and easier to debug as it is only a single stream of data. The simple MLP is a scaled down version of the main MLP structure containing many of the same biasing cells but does not have any of the switch cells. The basic biasing scheme flows as follows: first, the master bias current is sent on-chip; next, the master bias is mirrored to the bias control circuitry that controls whether currents are sent to the neuron/WTA blocks; lastly, the bias current is sent and mirrored into the bias cells based upon how many of the current mirrors are programmed in each neuron/WTA block. The bias programming is controlled by a string of shift registers that are made up of basic D-type Flip-Flops, which an example of a 4-bit shift register can be seen in Figure 3.8. Figure 3.9 shows the master bias input current structure and one of the current mirrors that would be controlled to send current to a single neuron or WTA block. Figure 3.10 details the bias input structure for the neuron as well as a single current mirror for the biasing of the neuron or WTA block. The programming of the neuron or WTA biasing determines the number of current mirrors in parallel for each bias current input. Finally, each neuron block in the simple MLP only contains one multiplier and one sigmoid circuit.

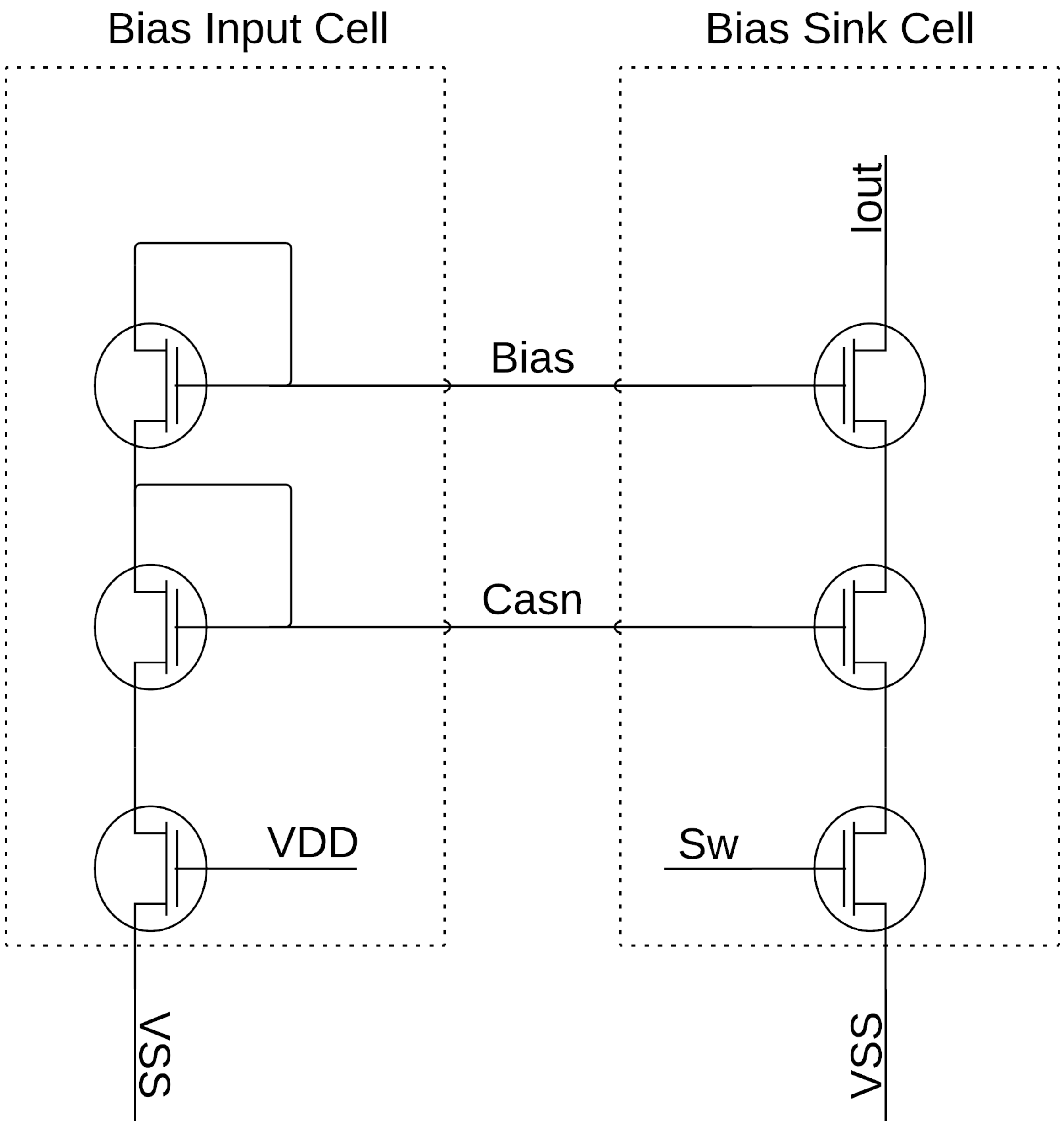


Figure 3.9: Master Bias Input Cell with Single Current Mirror Output.

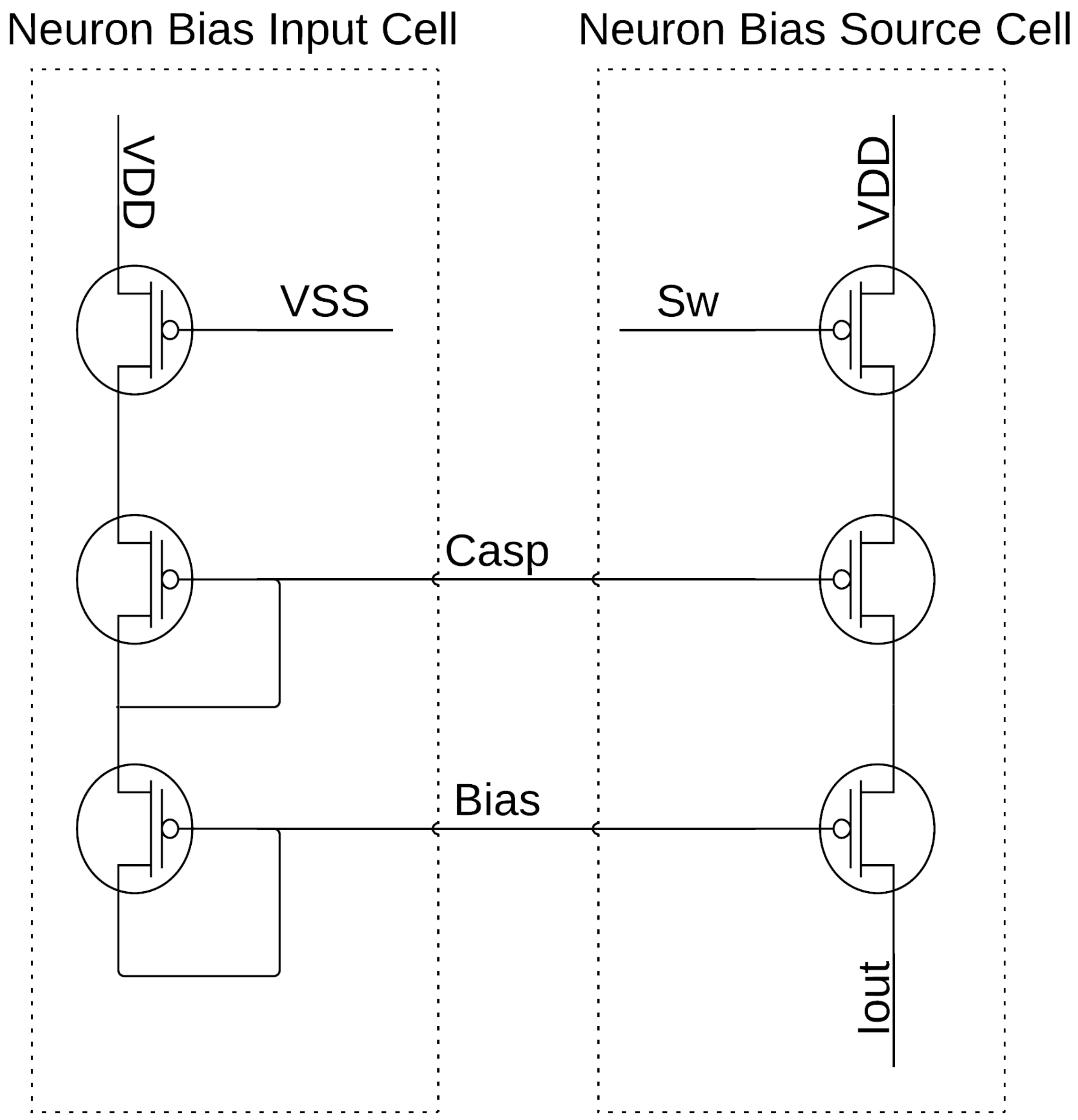


Figure 3.10: Neuron/WTA Input Bias Cell with Single Current Mirror Output.

The main MLP system in Figure 3.7 consists of twelve neurons, two WTA blocks for the main system outputs (shown in Figure 3.4), S-switch matrices, and C-switch matrices. Starting with the neurons, each contains four multipliers and a single sigmoid which is shown in Figure 3.11. Normally, a neuron for a configurable MLP would have a much larger number of multipliers per neuron as each neuron should be capable of receiving inputs from every other neuron in the previous layer. The total inputs are limited to four per neuron as it would be impractical and highly disadvantagous to circuit operation to have a large number of multipliers in each neuron in addition to limiting the amount of interconnects and pads required (as this design is limited by the number of pads and interconnects available due the space constraints and process technology, respectively). The constraint on the inputs maintains signal integrity by decreasing routing and switching characteristics that would be needed for a higher number of multipliers as well as helps constrain the amount of chip area required for the main MLP system as a whole. Much like the simple MLP's neurons, the main MLP's neurons have several tens of current mirrors and shift registers for biasing and configurability, respectively.

The next MLP structures that will be discussed are the two switch matrices. The S-switch and C-switch matrices are developed from the floating gate switch cells in [25] and

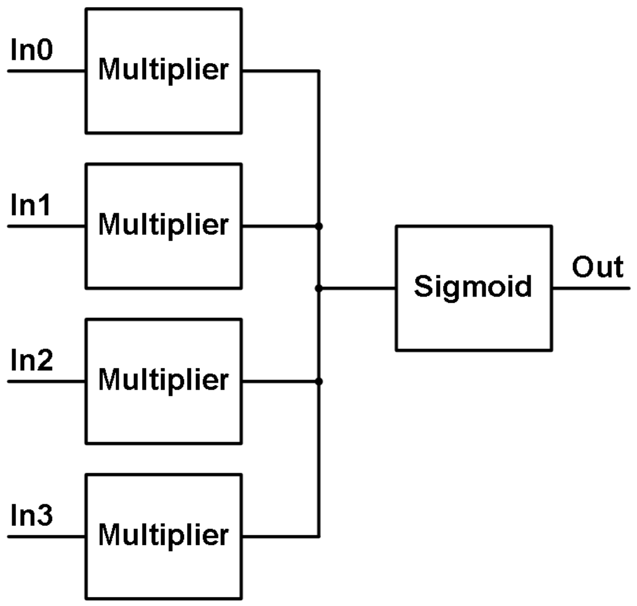


Figure 3.11: Main MLP Neuron Block Diagram.

take the form of the switches in Figure 2.27e and 2.27d, respectively. The main difference being that the switches in the main MLP's matrices are a single PMOS transistor whose gate is controlled by the output of a shift register instead of a floating gate node. Floating gate switches were initially considered for the switch matrices but resulted in too much programming burden in addition to inconsistentcies in how each gate is initialized during fabrication for the large number of switches needed for the MLP signal routing. The basic C-switch matrix has eight vertical routes that can be connected to five horizontal routes (four for neuron inputs and one for the output). Figure 3.12 shows these vertical and horizontal routes with the C-switch (single transistor) linking them together when activated. The S-switch matrix allows the ability to route a signal north, south, east, or west with as few switches as possible and can be seen in Figure 3.13. Each S-switch construct contains six transistors for the desired signal connections and a 3-bit address decoder to simplify the number of shift registers required to program a single S-switch structure. Additionally, the S-switch matrix allows routing through a layer of neurons to the next layer if desired.

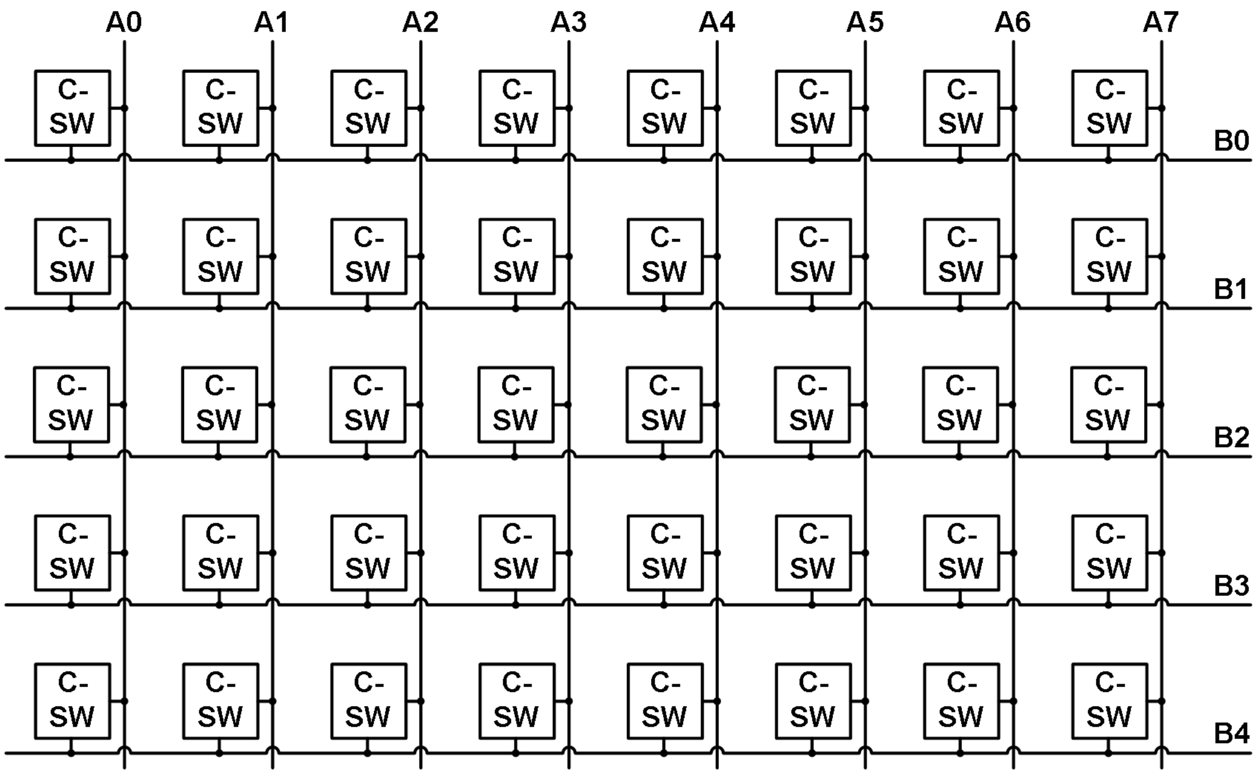


Figure 3.12: C-Switch Matrix.

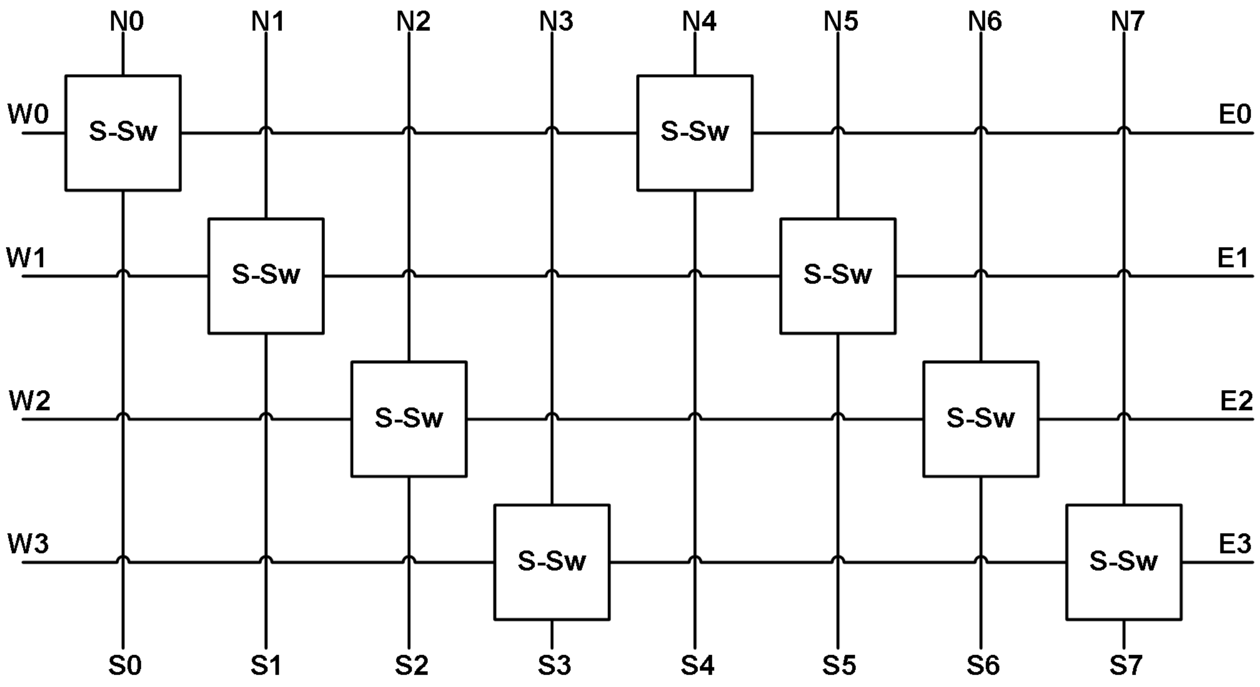


Figure 3.13: S-Switch Matrix.

The main MLP system operates by routing one or more of the input signals to the first layer of neurons. Next, those neurons will weigh their inputs at the multipliers, sum the multiplier currents together before the sigmoid input, and then output a signal to the next layer of neurons. This operation will continue until the desired number of layers is achieved or all resources on the chip are used. The last layer of neurons will output to one or both of the WTAs that will then amplify and compare the signal to a reference before outputting the digital version of the final signal. The system requires four data streams for switch programming, four data streams for neuron and WTA bias programming, and one data stream for master bias control programming. The configurability of the main MLP system allows the user to create a diverse range of basic MLP neural networks that are capable of low power usage at higher frequences in the MHz range.

3.5 Summary

In summary, the entire MLP system is developed utilizing and improving upon the previous works reviewed in Chapter 2. The multiplier circuit is designed for high speed current operation while maintaining a lower power state in the subthreshold region. The sigmoid circuit is designed for the same type of speed and power operation as the multiplier and creates the desired neural network function by applying the input signal to a logistic function. Thus, these two structures create the basic neuron that is copied throughout the MLP system along with the WTA blocks. The WTA block is designed to amplify an input signal through a multiplier, then compare the signal to a set reference current, and then create a digital voltage waveform from inverters and OR gates. Along with these vital structures, the MLP has several hundred switches configured with two different types of matrices for different switching operations. The switches are programmed through shift registers that are also copied and used for configuring the bias cells for each neuron and WTA, as well as the bias control circuitry. The final physical layout of the main MLP system can be seen in Figure 3.14 (area: 1 mm by 1 mm). Overall, the MLP system seems complex but consists of several simple circuits that are copied and pieced together strategically to form a configurable low power neural network.

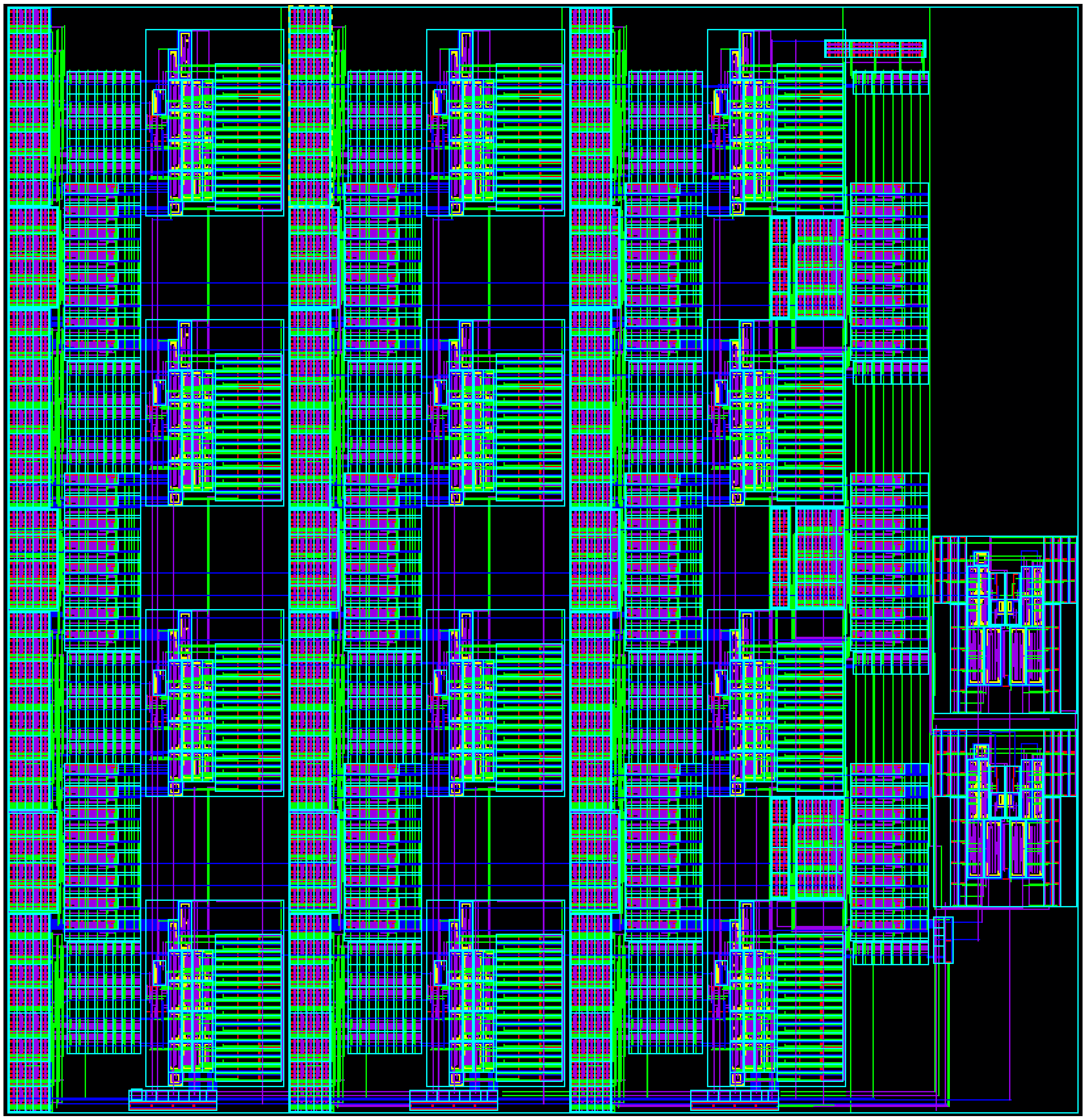


Figure 3.14: Main MLP System Physical Layout (Area: 1 mm by 1 mm).

Chapter 4

MLP Simulation and Measurement Results

4.1 Simulation Results

This section details the simulation results obtained from the Analog Design Environment (ADE) within the Cadence design suite. The simulation results show the behavior of the corresponding circuit under mostly ideal situations. Any deviations from ideal would be resultant upon the addition of parasitic resistances or capacitances and would be explicitly stated next to the non-ideal simulation results. Generally, the testbenches for these results utilize ideal voltage and current sources to provide the proper biasing and waveforms to test each circuit. The subsections are ordered similarly to Chapter 3 and refer to the final designs discussed in each section of the previous chapter. Finally, all the simulation results are performed with input signals in the ones of Mega-Hertz (MHz) range instead of the much higher frequency range (100's of MHz range) that the circuits are designed to operate within in order to mimic the frequencies of the physical test board whose limitations will be discussed in the next section. All simulations use a V_{DD} of 1.2 V and a V_{SS} of ground (0 V) unless otherwise specified.

4.1.1 Multiplier

The multiplier design, discussed previously and shown in Figure 3.1, is verified for correct functionality using a testbench that provides three DC currents in addition to the input current signal. For better reference, the multiplier design is shown here again in Figure 4.1. The three DC current sources provide the bias current (I_b), the weight current (I_2), and the saturation current (I_3) for this multiplier cell. The bias current has a value of 25 nano-Amps (nA) while the weight and saturation currents both have an initial value of 100 nA for the first simulation. The input current signal I_1 is a pulse train consisting of ten pulses at a frequency of 1 MHz with amplitudes of 100 nA for the on state and zero current for the off state. The output current signal I_4 is connected to a sigmoid input to provide the typical load seen by this design. The first simulation run is shown in Figure 4.2.

Figure 4.2 consists of six waveforms representing four signals. Going from top to bottom, the first two waveforms reflect the voltage levels at the input I_1 and output I_4 nodes as the input current signal changes. The voltage for the input signal peaks at roughly 1.03 Volts as this value represents the 100 nA amplitude. The minimum voltage level for the input occurs when the current is zero and is represented by any value less than approximately 750 milli-Volts (mV). The second voltage waveform details the voltage levels as the output sinks the final current value from the connected sigmoid circuit input (diode-connected PMOS transistor for reference). For this voltage signal, the minimum value occurs when the output

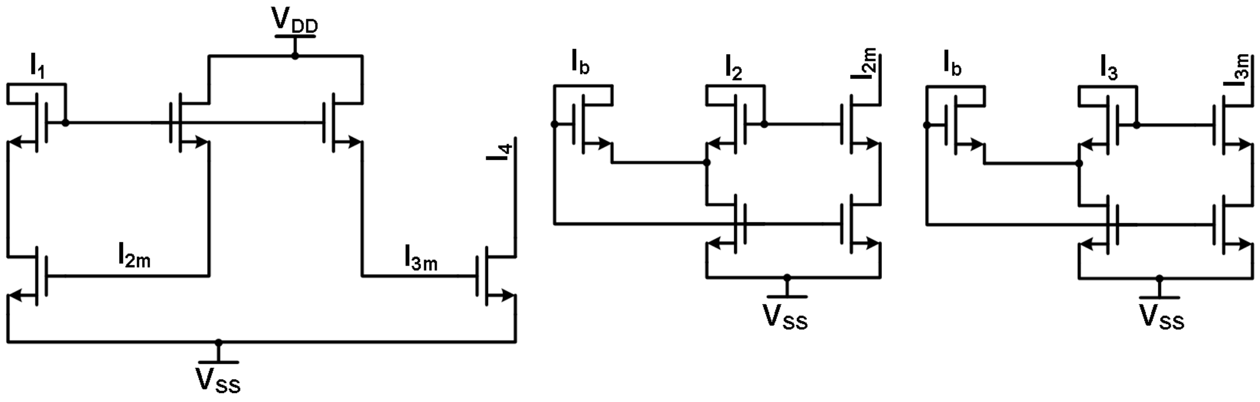


Figure 4.1: Multiplier Circuit Schematic.

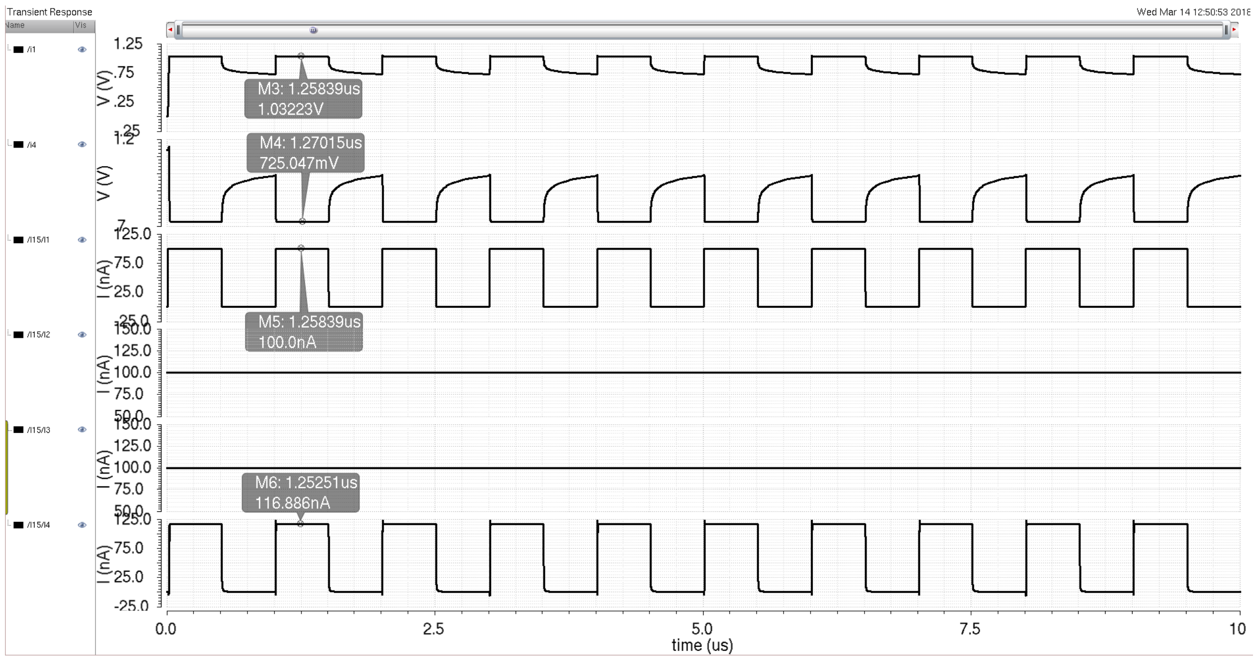


Figure 4.2: First Multiplier Simulation Run.

current is a maximum. The bottom four waveforms detail the four current signals associated with the multiplier.

The input signal I_1 (third from the top of Figure 4.2) can be seen pulsing at 1 MHz with a 100 nA amplitude as described previously. The next two current signals (I_2 and I_3, respectively) do not fluctuate as they remain at 100 nA for this simulation. The final current signal at the bottom of the figure is the output current I_4 which mimics the input signal with a frequency of 1 MHz but has an amplitude of approximately 116 nA. The output amplitude difference is seen because of slight variations in well-modeled transistors in the subthreshold region. Additionally, the output node has a single device between it and its respective current source resulting in more voltage headroom for the output device whereas the input node has two devices in series allowing for less headroom for each. The expected output for this simulation run is a pulse train with an approximate current amplitude of 100 nA for the I_4 signal, which is consistent with what is seen in Figure 4.2.

The next four figures (Figures 4.3 through 4.6) represent the multiplier's behavior in different bias conditions for the saturation current. The waveforms represent the same signals as in Figure 4.2 and are in the same positioning as well for easy comparision. Figure 4.3 is the multiplier circuit with the saturation current I_3 doubled to 200 nA that reduces

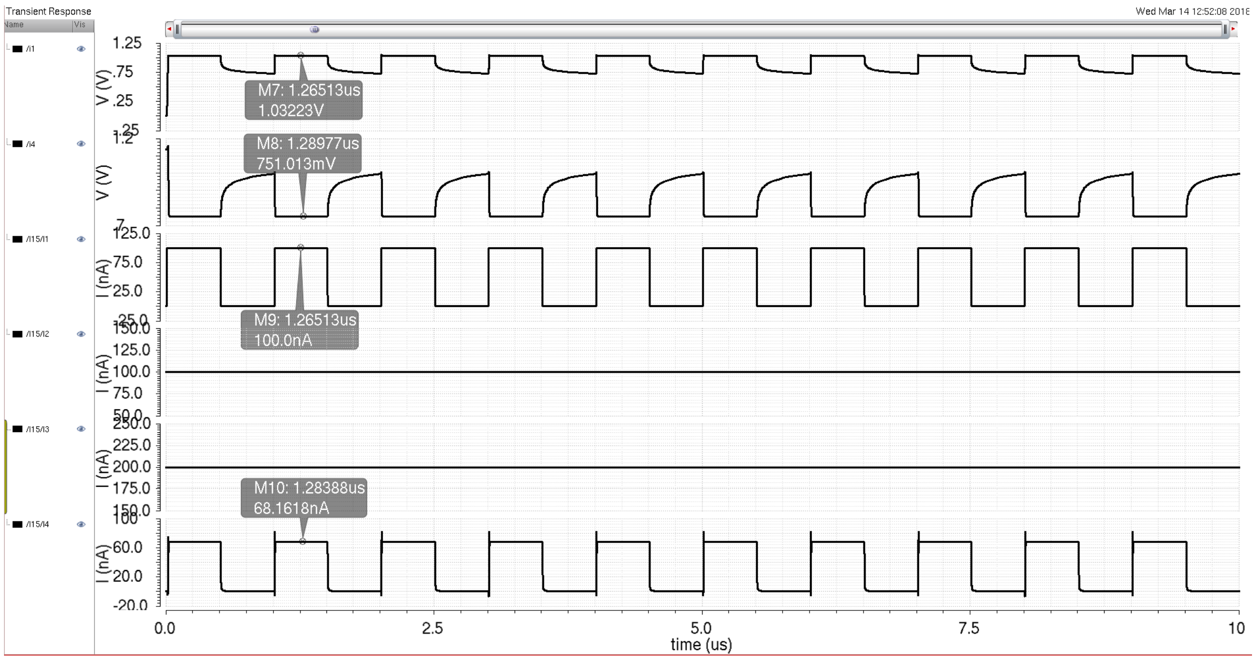


Figure 4.3: Second Multiplier Simulation Run.

the output current amplitude to roughly 68 nA, which is close to the expected output of 50 nA considering the voltage headroom differences previously mentioned. Figure 4.4 shows the multiplier operating with the saturation current I_3 quadrupled (400 nA) with respect to the first simulation run that reduces the output current amplitude further to approximately 34 nA, which is again close to the expected 25 nA. Figure 4.5 demonstrates multiplier functionality when the saturation current is halved to 50 nA with regards to the first run producing an output current amplitude of around 181 nA, which shows the expected functionality of almost doubling the output current. Lastly, Figure 4.6 details the multiplier's output current at an amplitude of 266 nA for the case of the saturation current being 25 nA. This final simulation is expected to have an output current amplitude of 400 nA, but the limits on voltage headroom limit the multiplier's functionality. These four test cases are examples of the many different ways for the multiplier to weight its output current amplitude in order to produce the desired signal.

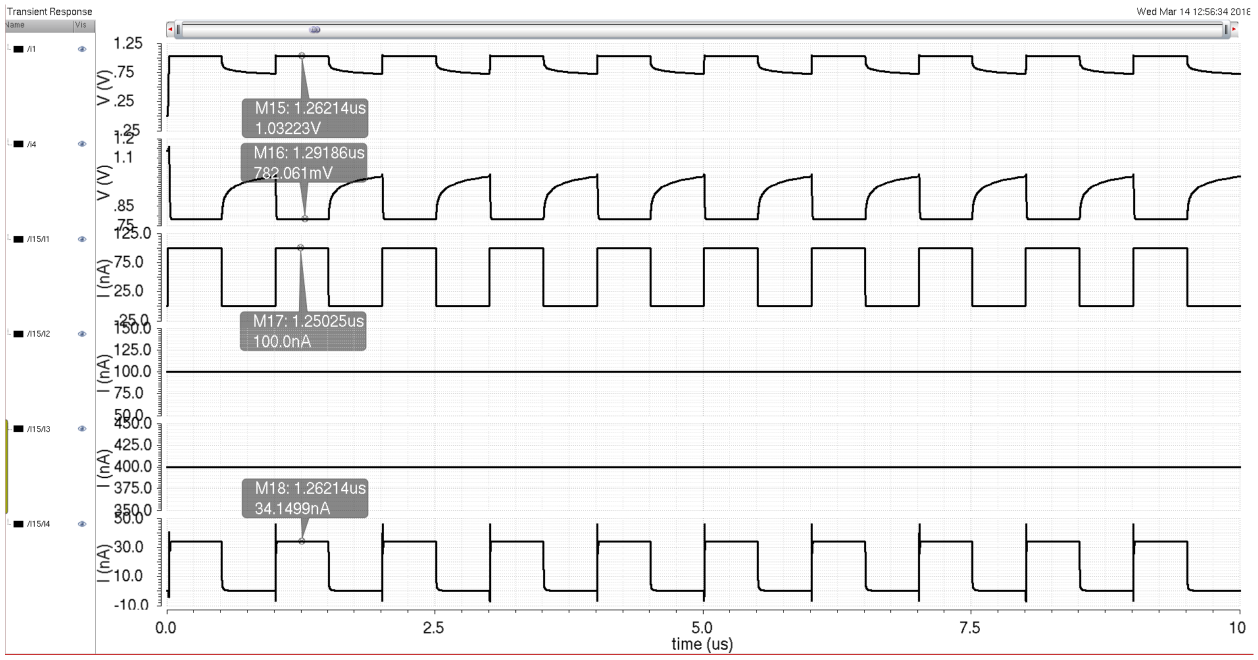


Figure 4.4: Third Multiplier Simulation Run.

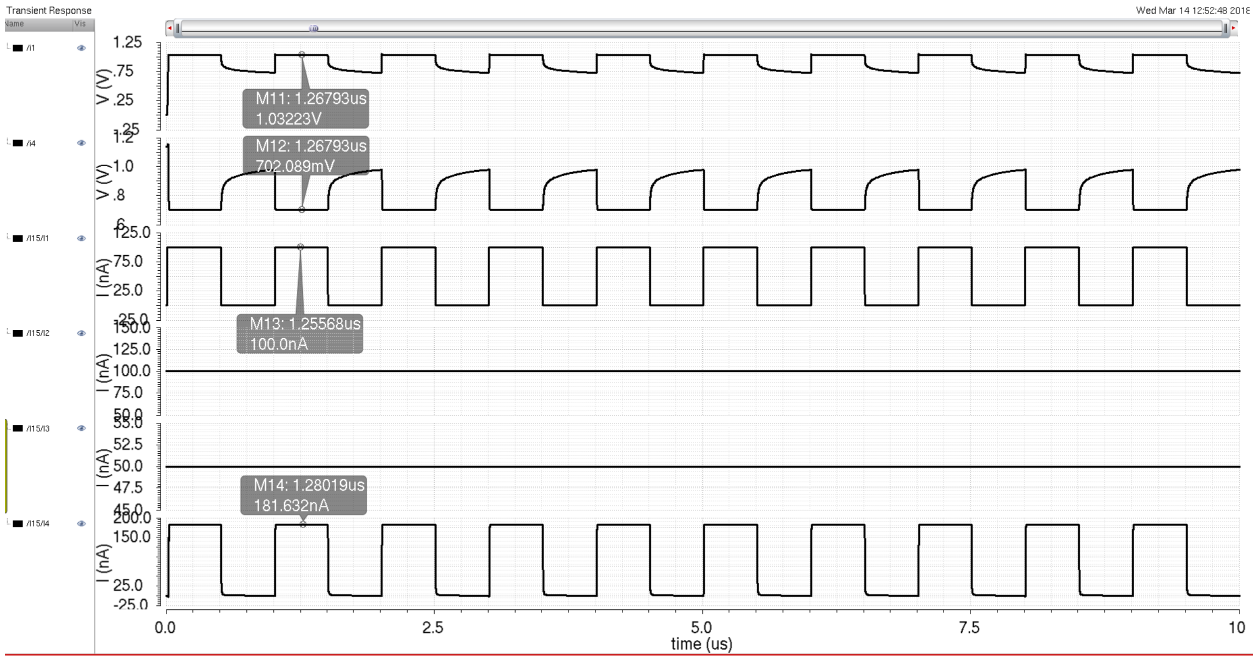


Figure 4.5: Fourth Multiplier Simulation Run.

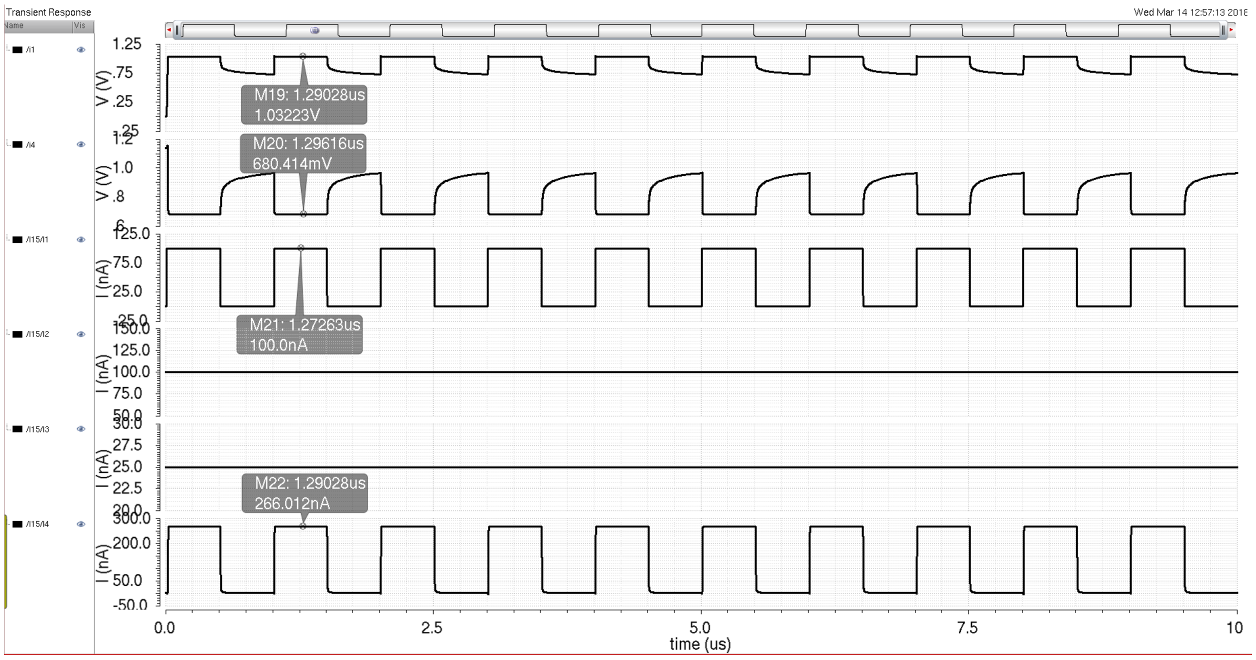


Figure 4.6: Fifth Multiplier Simulation Run.

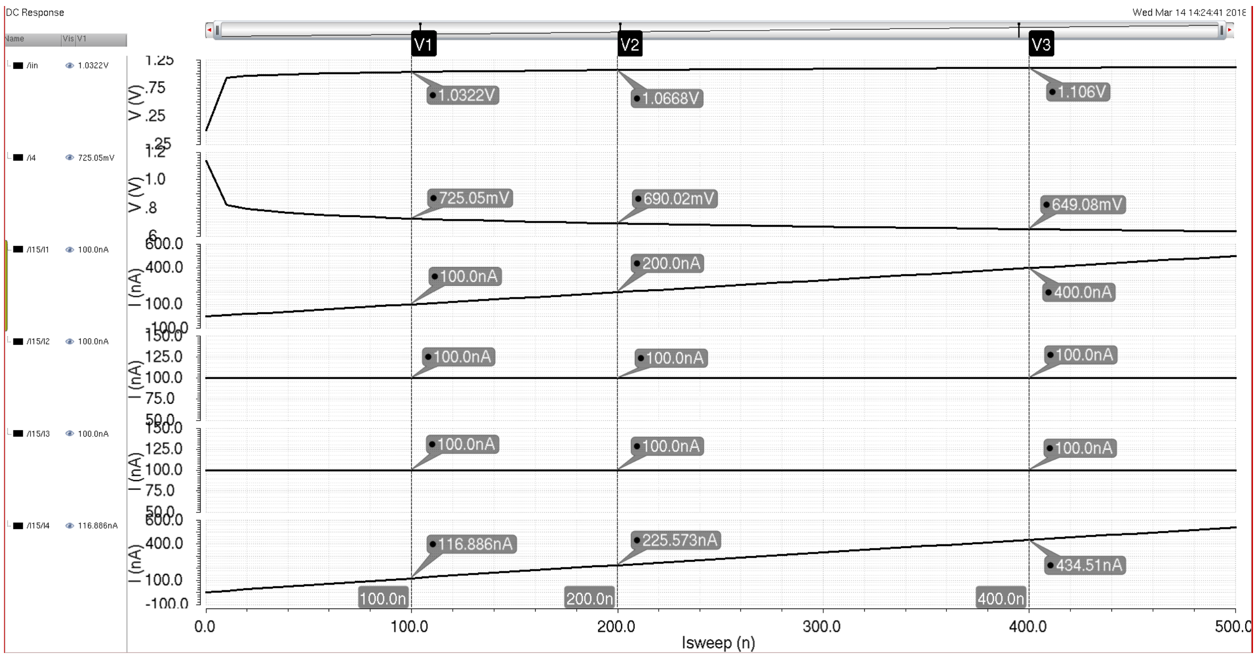


Figure 4.7: Multiplier Input Signal DC Sweep Simulation.

The final simulation figure for the multiplier is a DC sweep of the input current signal I_1 and can be seen in Figure 4.7. The simulation has the same six waveforms for the four current signals in the same position in the figure as the previous simulation results. The purpose of the DC sweep on the input signal is to show how the input signal's current level affects the output signal I_4 current level as well as the voltage levels for both signals. Figure 4.7 shows that as the input current increases the output current does follow it almost linearly. However, the output current's percent difference decreases as the current increases. This effect is due to the more and more gradual decrease in the output signal's voltage level with respect to current. Similarly, the input signal's voltage level also increases more gradually as the current increases. These effects are due to the transistors in the input and output nodes requiring more voltage headroom to provide more current and are quite similar to the I-V curves seen in electronics textbooks.

In conclusion, the five transient and single DC sweep simulations verify the multiplier's capability to function efficiently and sufficiently accurate while adding user controlled signal weighting in the overall MLP system.

4.1.2 Sigmoid

The final sigmoid design is verified in a similar fashion much like the multiplier is in the previous section. For reference and clarity, the sigmoid design from Chapter 3 is shown again in Figure 4.8. The sigmoid circuit has four current signals with two being DC currents and two being input/output signals. The testbench for the sigmoid consists of three current sources for the three current inputs. The first two current inputs are DC signals for the bias current I_b and the reference current I_n. The current values for the bias current and reference current are 200 nA and 25 nA, respectively. The input current signal I_p is the same current pulse train used for the multiplier. Reiterating, the pulse train has ten pulses at a frequency of 1 MHz with an amplitude of 100 nA. The difference between the multiplier and sigmoid pulse train current source is that the sigmoid one is a current sink rather than a current source. Finally, the output current signal I_{out} will detail the logistic function output while being loaded with the multiplier input signal node. This load represents the typical next stage circuitry seen by the sigmoid design.

Figure 4.9 shows the simulation results for the sigmoid verification. However, going from top to bottom, this time the first four waveforms are the current signals and the last two are the voltage levels for the input and output switching currents. The first signal is the input current signal I_p and has the frequency and amplitude characteristics described in the previous paragraph. The next two signals are the reference current and the bias current with their levels being at 25 nA and 200 nA, respectively. The third signal from the bottom and last current signal is the output current I_{out} which has a matching frequency of the input's of 1 MHz and has an amplitude of approximately 94 nA. The two reasons why this current does not reach the maximum theoretical value of the bias current are that again there is not enough voltage headroom with the multiplier input node as a load and the input signal has not reached the saturation point in the sigmoid's logistic function (which will become apparent later on in this section). Finally, the last two waveforms depict the voltage levels for the on and off states of the input and output current signals, respectively. This simulation is expected to produce an output current pulse train with some degradation due to the sigmoid nature of the circuit at the edges of the pulse, which is consistent with what is seen in Figure 4.9.

The next simulation results are obtained from performing a DC sweep on the input current signal I_p while maintaining the same current levels for the reference and bias currents.

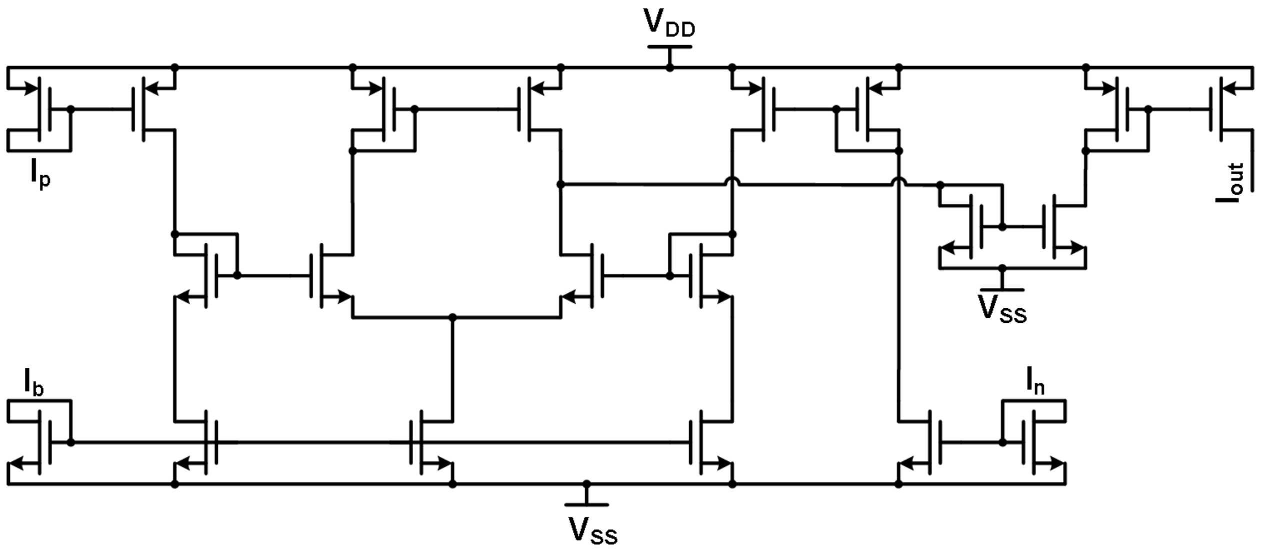


Figure 4.8: Sigmoid Circuit Schematic.

This DC simulation is expected to show a sigmoid-like waveform for the output current as the input is swept linearly. The results are depicted in Figure 4.10 and have the same order and positioning as the waveforms in Figure 4.9. The input current is swept from 0 to 500 nA and only the bottom three signal waveforms alter with the sweep. The second signal from the bottom is the voltage level for the swept input current. This voltage details a diode-connected transistor's voltage increase as more current is sourced from it and is similar to the curves seen in the DC sweep for the multiplier. The signals third from the bottom and at the bottom depict the behavior of the output current signal and output voltage signal, respectively. Both waveforms take the shape of a logistic function which is the overall goal of the sigmoid circuit. As mentioned earlier, the current level for a 100 nA input signal is below the saturation point of the logistic curve and is only at about 94 nA. The curve shows that it would take an input signal around 400 nA to be securely in the saturated section of the logistic function.

In conclusion, the sigmoid design is verified through a transient and DC sweep simulations. The results demonstrate functionality at the desired frequency with the ability to adjust the output current levels depending upon the input current I_p and the bias current

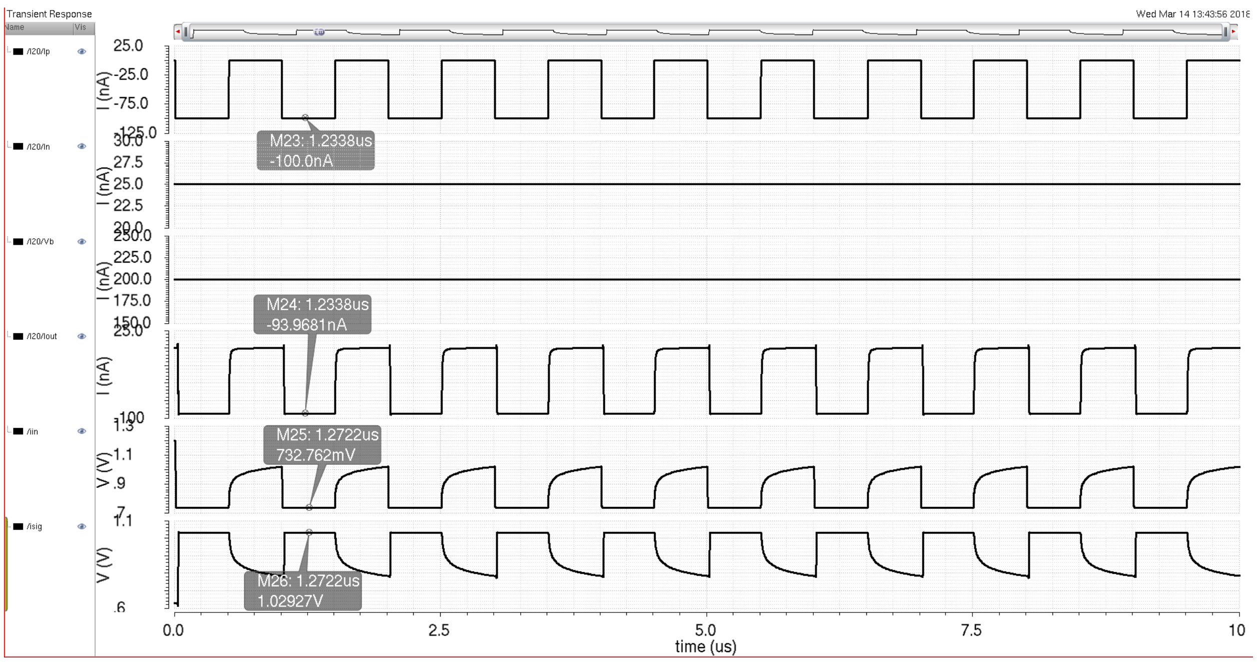


Figure 4.9: Sigmoid Simulation Run.

I_b as well as the reference current I_n. All three adjustable currents give greater user control at the MLP system level.

4.1.3 Thresholding Circuit

The thresholding circuit (TC) final design is tested using the same methods as the previous two circuits and has its circuit shown again for reference in Figure 4.11. This figure is only the TC cell but keep in mind that there is a multiplier cell that is inline prior to every TC cell. In addition to the multiplier cell, the TC cell has six biasing currents as well as the input current signal and an output voltage signal. Three of the biasing currents are Minch cascode bias currents much like I_{b1} in Figure 4.11 with all having current values of 25 nA. The other three biasing currents are the weight current (I_w on the simulation), the saturation current (I_s on the simulation), and the TC reference current (I_b on Figure 4.11). The input current signal I_{in} on the schematic is the intermediate current signal that is output from the inline multiplier used for scaling the current signal. After the current comparison via the two current mirror networks, the output voltage V_{out} is buffered through the two inverters and either goes to the 2-input OR gate or off-chip for analysis.

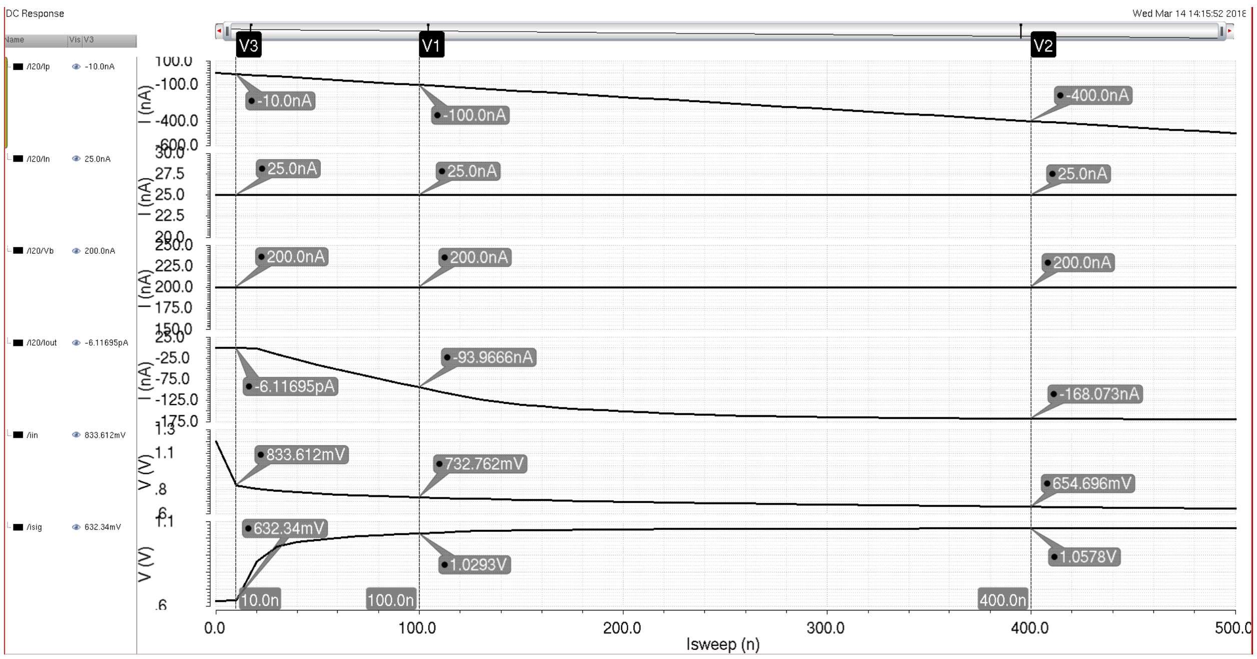


Figure 4.10: Sigmoid Input Signal DC Sweep Simulation.

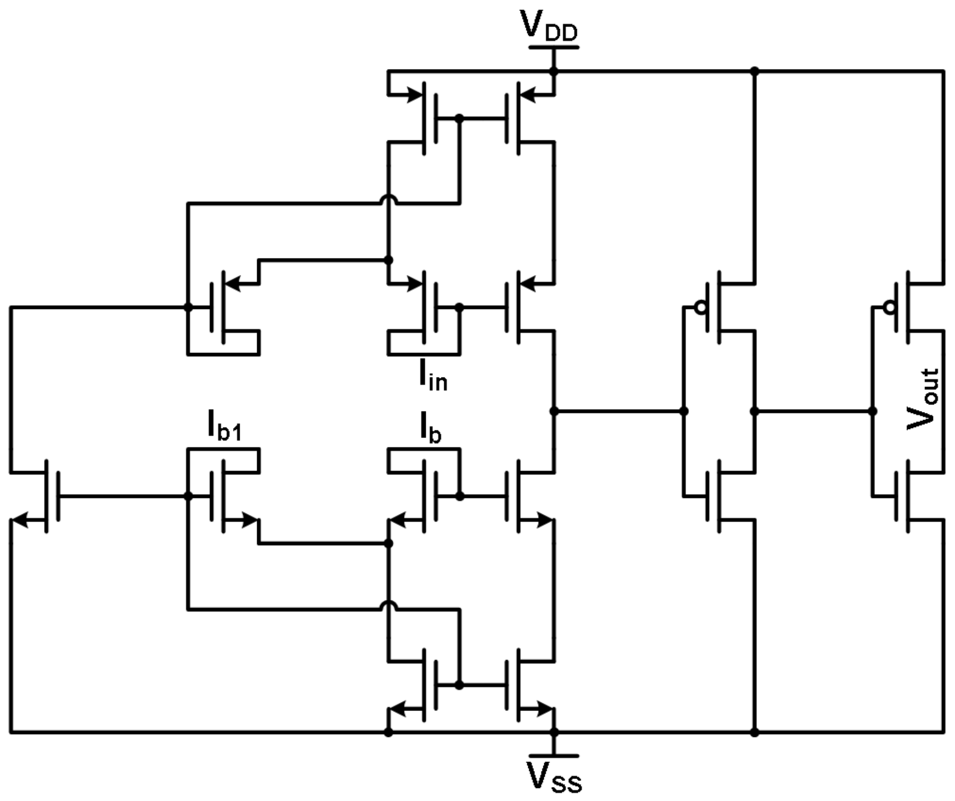


Figure 4.11: Thresholding Circuit Cell Schematic.

Figure 4.12 details the transient functionality of the WTA block (with the inline multiplier). The expected output waveform is a digital voltage pulse train from 0 to 1.2 V corresponding to the input current "high" and "low" states. The simulation figure consists of seven waveforms that make up the five important current signals and the output voltage. Starting with the topmost current signal (third waveform from the top), this signal is a 1 MHz pulse train current source with an amplitude of 100 nA and ten pulses. The next three current waveforms moving downward are the weight current, the saturation current, and the reference current which all have a current value of 100 nA. The bottommost waveform is the intermediate current signal from the output of the multiplier to the input of the TC cell. This signal, labeled I_{in}, mimics the input current signal in terms of the 1 MHz frequency and has a slightly scaled amplitude of 110 nA (due to voltage headroom differences between output and input nodes in the multiplier that are discussed previously in this chapter). The two topmost waveforms are both voltage signals which consist of the voltage created by the input current signal at the multiplier's input node (topmost) and the final output voltage

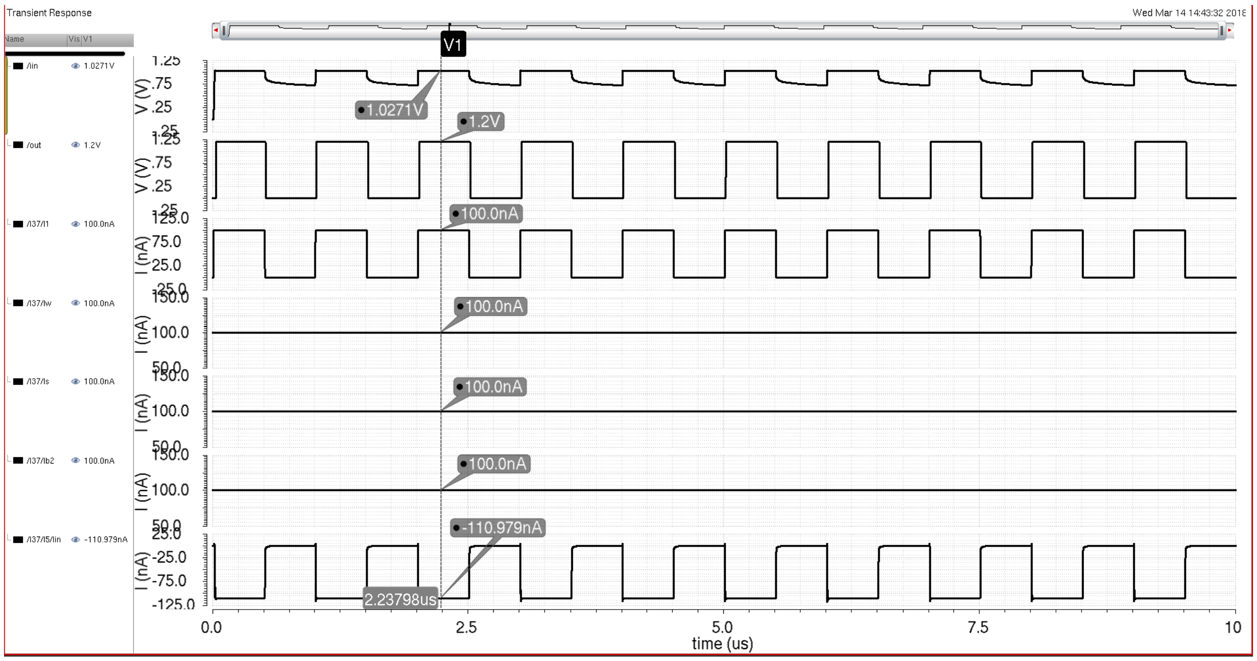


Figure 4.12: Thresholding Circuit Simulation Run.

signal that is the desired output for WTA block (second from the top). The output signal V_{out} depicts the correct functionality of the TC cell as the waveform has the 1 MHz frequency from the input current signal as well as has been succesfully converted to a digital output voltage to easy analysis or observation off-chip.

The final two simulation figures for the WTA block demonstrate the relationship between the output voltage V_{out} and the input current I_{in}. The expected behavior of these simulations is that the output signal should switch from a "high" to "low" state or vice-versa depending upon the simulation when the threshold at the switching node is passed. Figure 4.13 shows how performing a DC sweep on the input current signal changes the output voltage. In this figure, the waveforms maintain the same order and positioning as in Figure 4.12. The main relationship to point out from this simulation is that the input current signal only requires 40 nA to change the output voltage when compared against a 100 nA reference current. Before analyzing this relationship further, a look at Figure 4.14 can reinforce the outcome seen in Figure 4.13. This second DC sweep simulation is performed by sweeping the reference current in the WTA cell. The sweep details that for a 100 nA constant input current signal a reference current of 270 nA is required to flip the output voltage from high to low. This relationship requiring more pull-down current than pull-up current in the current comparison between the current mirrors is because of the circuit's design. In the current mirror in Figure 4.11 that reflects the Minch bias current from the NMOS to PMOS devices, the same voltage headroom is not maintained creating a much higher Minch bias current in the PMOS devices allowing them to operate better with less current than the NMOS devices. This behavior does not negatively affect the circuit and can be accounted for by increasing the reference current to counteract the input signal as needed.

In conclusion, the thresholding circuitry operates as expected and is verified through the simulations performed. Again, the TC results show the ability of the circuit to operate at the desired frequency range while maintaining a proper digital output voltage. The TC circuit is the final crucial component in the MLP system producing the final output of the whole system.

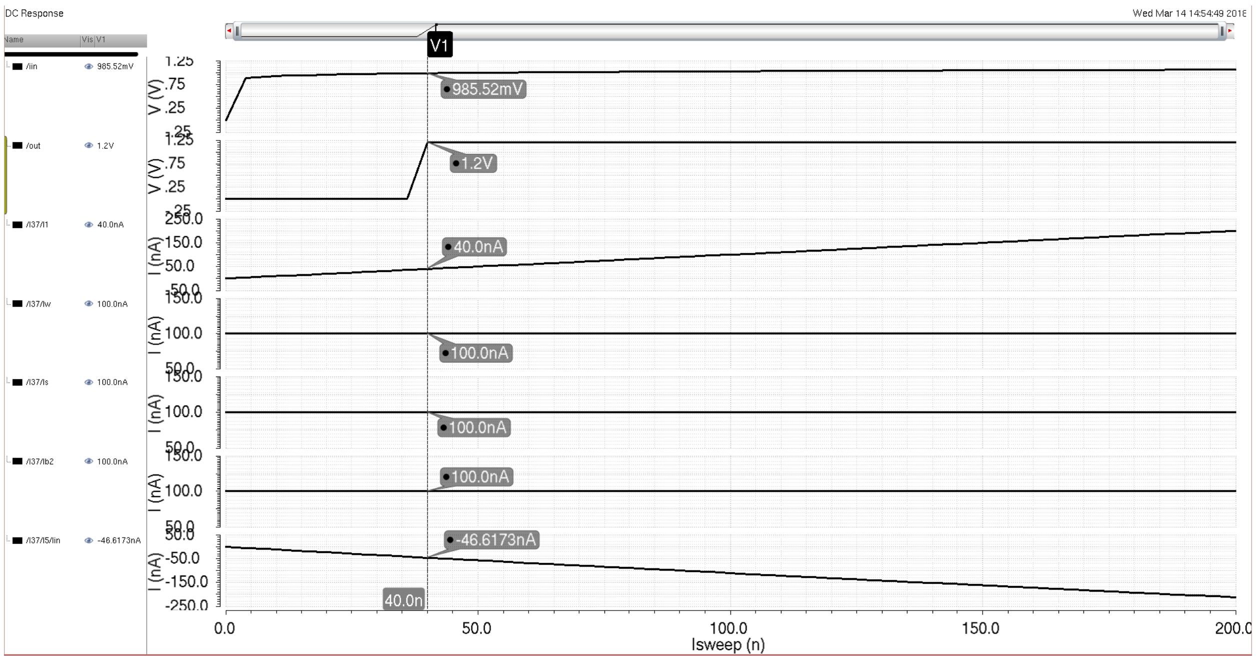


Figure 4.13: Thresholding Circuit Simulation Sweeping the Input Current Signal.

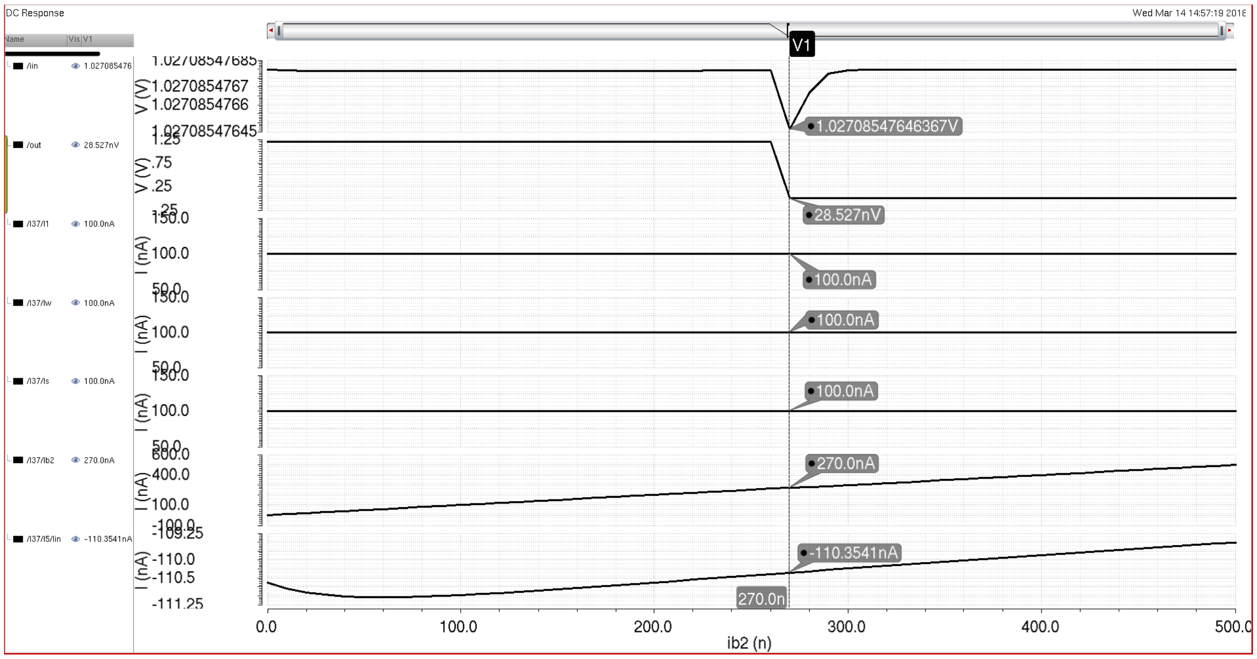


Figure 4.14: Thresholding Circuit Simulation Sweeping the Reference Current Signal.

4.1.4 MLP System

The final simulation to analyze is the main MLP system. The simulation is performed with the MLP programmed to one of its many configurations. For reference, the main MLP system block diagram is shown again in Figure 4.15. This simulation configuration utlizes 25% of the total assets in the main MLP system at the lowest programmed biasing and can been seen in Figure 4.16. For this simulation, the bias currents for the neurons and WTA have to be programmed using voltage sources and clock signals. Additionally, the switches connecting the inputs and outputs of the neurons together with other circuit structures have to be programmed in conjuction with the bias programming. This programming takes the majority of the simulation time and causes the simulation to be time consuming because of the transferral of data between the hundreds of shift registers. In addition, the simulation took into account the parasitics for the routing and layout of the physical integrated circuit. With these reasons in mind and the fact that the simulation took 8 days to finish, only one configuration is considered in terms of simulation results.

The MLP parasitic system simulation is shown in Figure 4.17 and only has four waveforms in it. The expected output is a digital voltage waveform that mimics the state of the input pulse train. Overall, the information from the simulation is simple and easy to parse with the complexity coming from programming the system to achieve correct functionality. The bitstreams used for programming the main MLP system can be found in Appendix A. The first waveform in Figure 4.17 is the current utilized by the system during operation from its single voltage rail. This current will be analyzed in the measurement results section to determine each configuration's figure of merit (FOM). For this section, it is used as a guideline to determine how successful the physical circuit will be. The second waveform is the voltage input signal that will produce the appropriate input current signal when applied to the multiplier in the first neuron. The 1.2 V amplitude produces an input current signal in the 100's of nA range. The third waveform is the digital output voltage for the system's first output when loaded with 12 pF capacitor which is similar to that of an oscilloscope probe. The last waveform is the digital output voltage for the system's second output and is a constant low as there is no signal routed to this output. The propagation delay times demonstrate the ability of the MLP system to produce the correct output after the input stimulus. The 637 ns value comes after the system has maintained a low setting which discharges the parasitic capacitors and inductors for longer whereas the 502 ns value comes after a quicker transition in which those parasitics are not allowed to discharge as much. Consequently for this load and programming, these delay values show that the maximum frequency that could be achieved with a successful output signal is around 15 to 20 MHz. The expected output is that the digital output voltage signal (third waveform) mimics the input voltage signal (second waveform) after some propagation delay caused by the system, which is what is seen in Figure 4.17.

In conclusion, the MLP system simulation details the entire system functioning as a neural network with all the neuron and system support circuitry working correctly. Even though the system appears to have high propagation delays, those values are only for a single configuration at the lowest bias settings for every circuit. Therefore, the ability to program different configurations and increase bias settings creates a system that can adapt to higher frequency demands to provide a proper output signal.

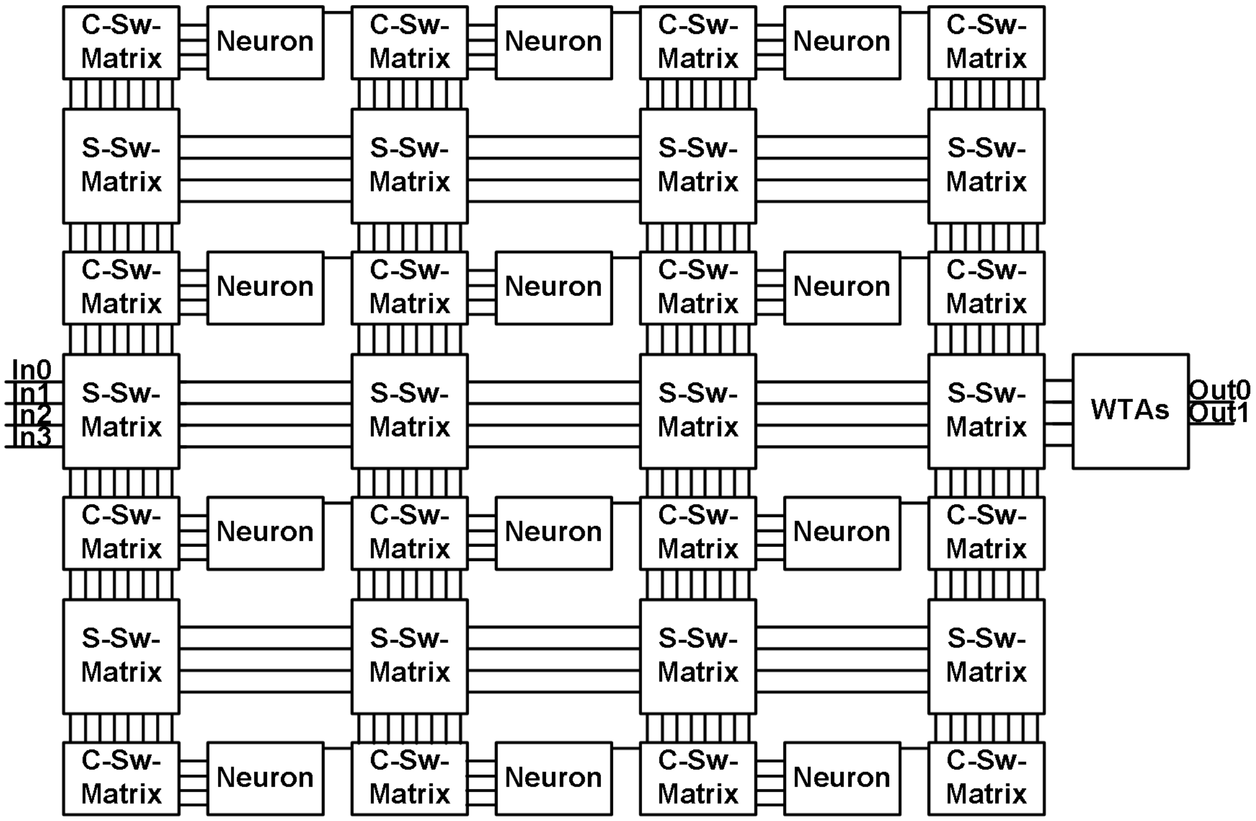


Figure 4.15: MLP Block Diagram for Configurable Low Power System.

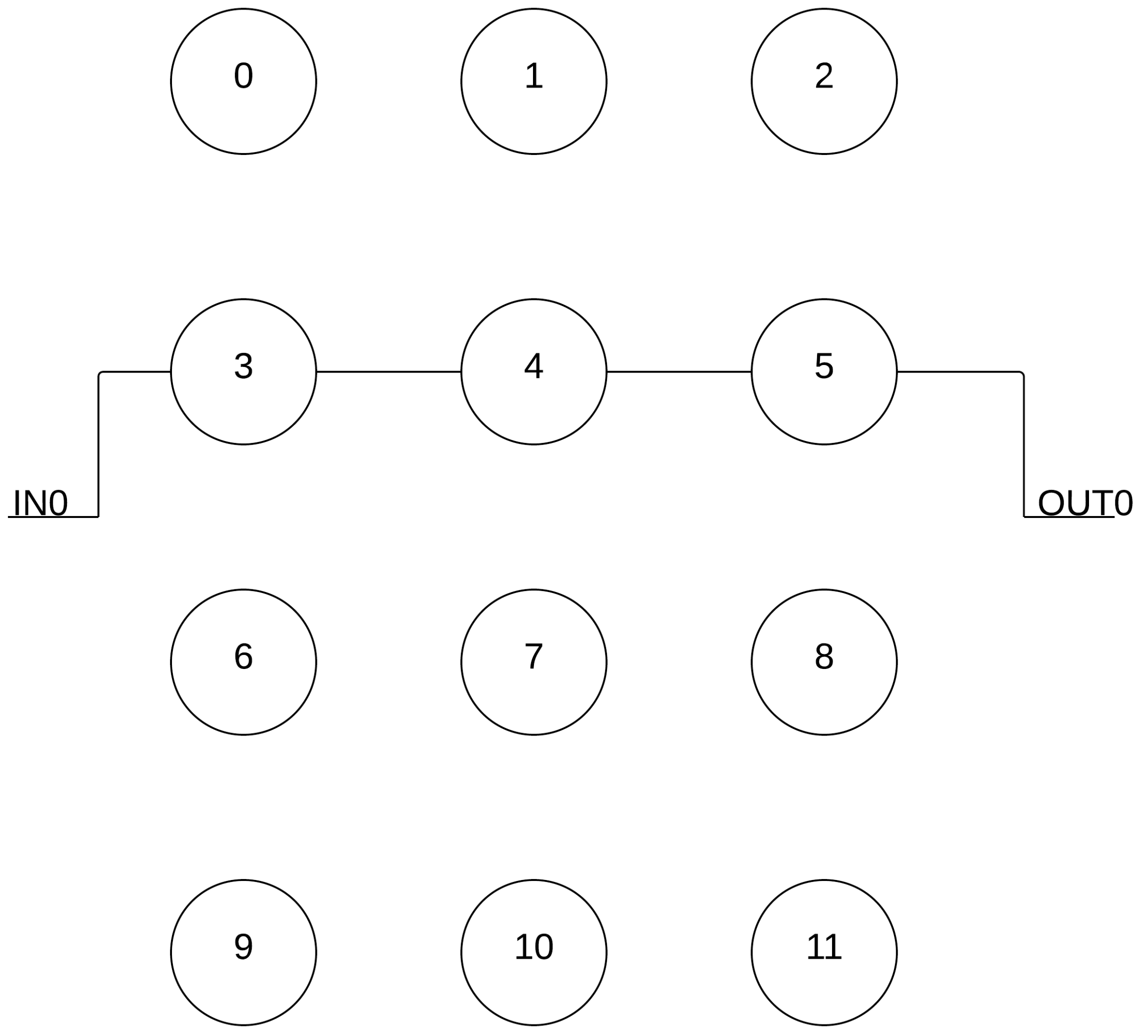


Figure 4.16: MLP Configuration for System Simulation.

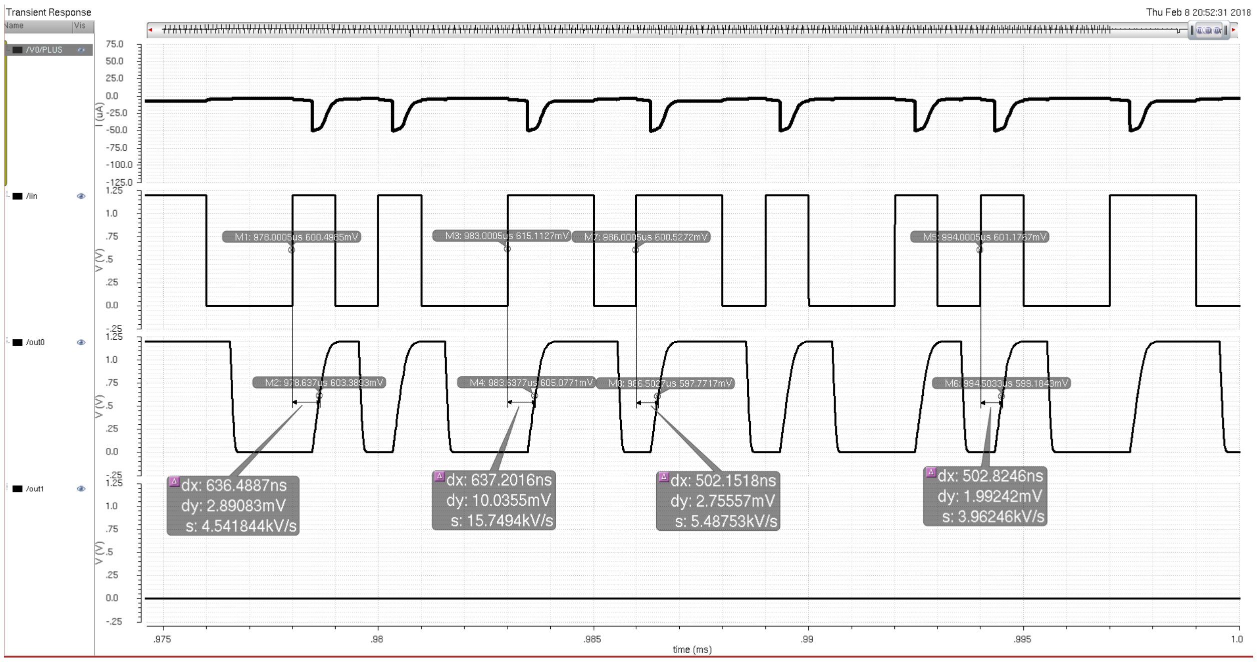


Figure 4.17: MLP Parasitic System Simulation.

4.2 Measurement Results

This section details the measurement results obtained from physically testing the manufactured integrated circuit on the printed circuit board (PCB). The PCB is shown in Figure 4.18 and contains circuits to test the different elements of the MLP system. The test board is divided into six sections: power circuits (red), microcontroller and supplemental circuitry (yellow), input signal circuits for four main inputs (purple), input signal circuits for simple MLP system (orange), input circuits for test structures (black), and socket for integrated circuit (white). These six sections allow every circuit to be thoroughly tested and analyzed. All power rails for the integrated circuit are kept at 1.2 V whereas the rails for the microcontroller are held at 3.3 V. The bitstreams and input signals are generated by the microcontroller, and the code for the microcontroller can be found in Appendix A. The measurement sections mirror that of the simulation sections except that there are no WTA test structures on the integrated circuit. A buffer circuit is used to obtain a cleaner output signal unless otherwise specified as the output structures of the chip are not strong enough to drive the 12 pF capacitive load of the oscilloscope probe at higher frequencies. In general, the PCB is capable of testing the main MLP system and other circuits at frequencies up to the ones of MHz. The limitations that cause this barrier are that the microcontroller unit can only produce reliable signals in this frequency range in addition to the added parasitic inductances and capacitances from the PCB traces. With these limitations in mind, the MLP system is still highly capable of reaching the desired figure of merit which will be shown in the MLP system later.

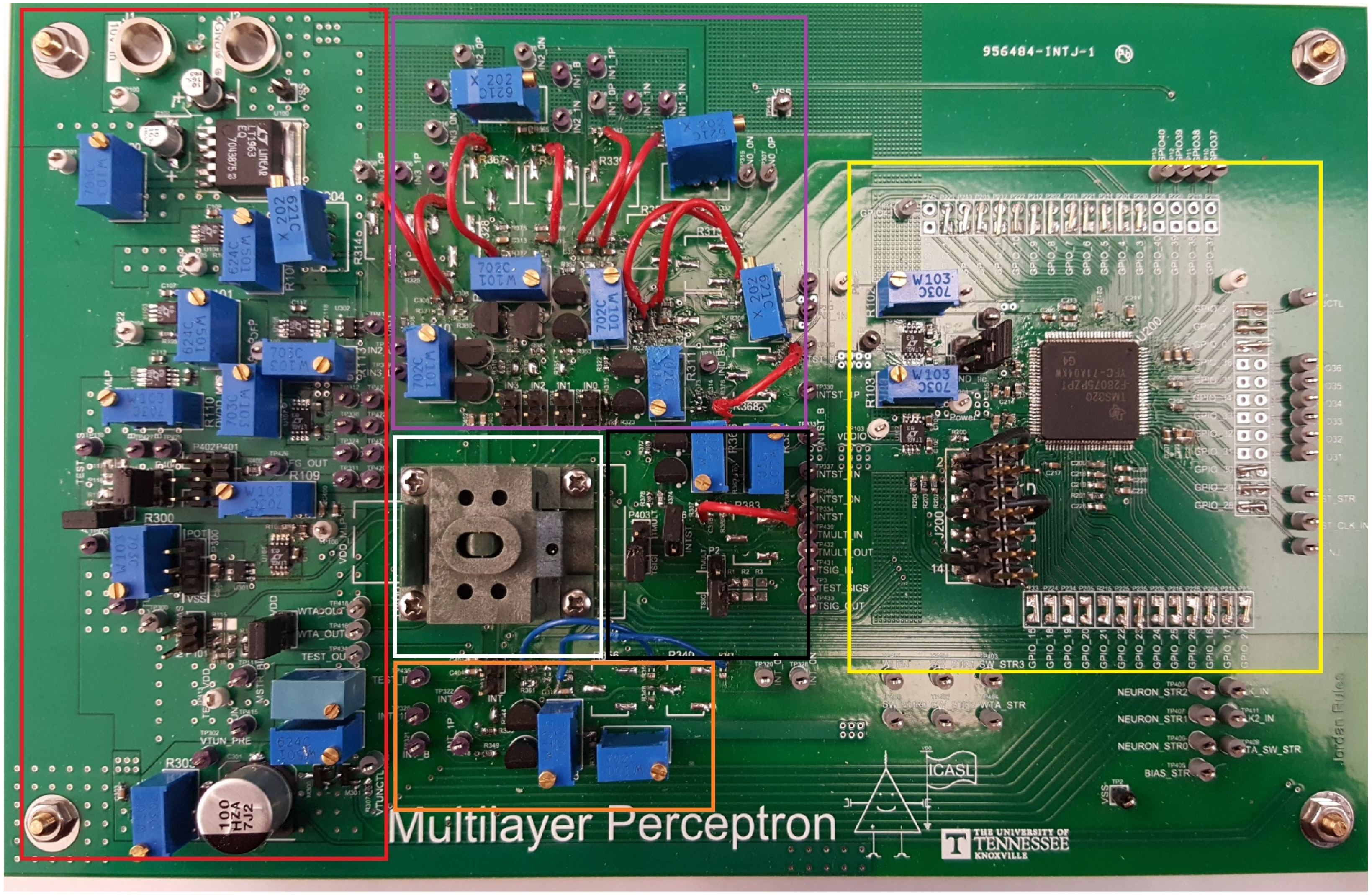
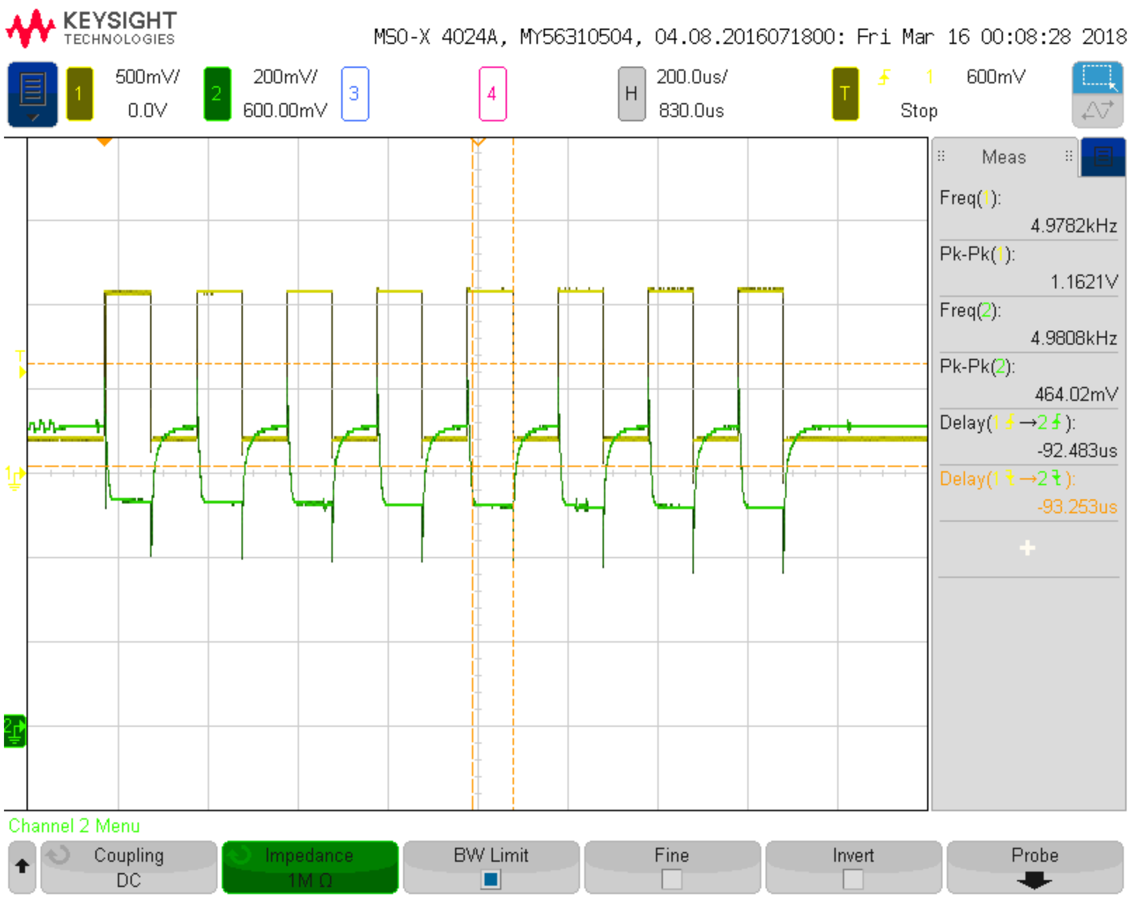


Figure 4.18: PCB Test Board for MLP Chip (Red = Power, Purple/Black/Orange = Input Signal Generation, Yellow = Microcontroller Unit, White = IC Socket).

4.2.1 Multiplier

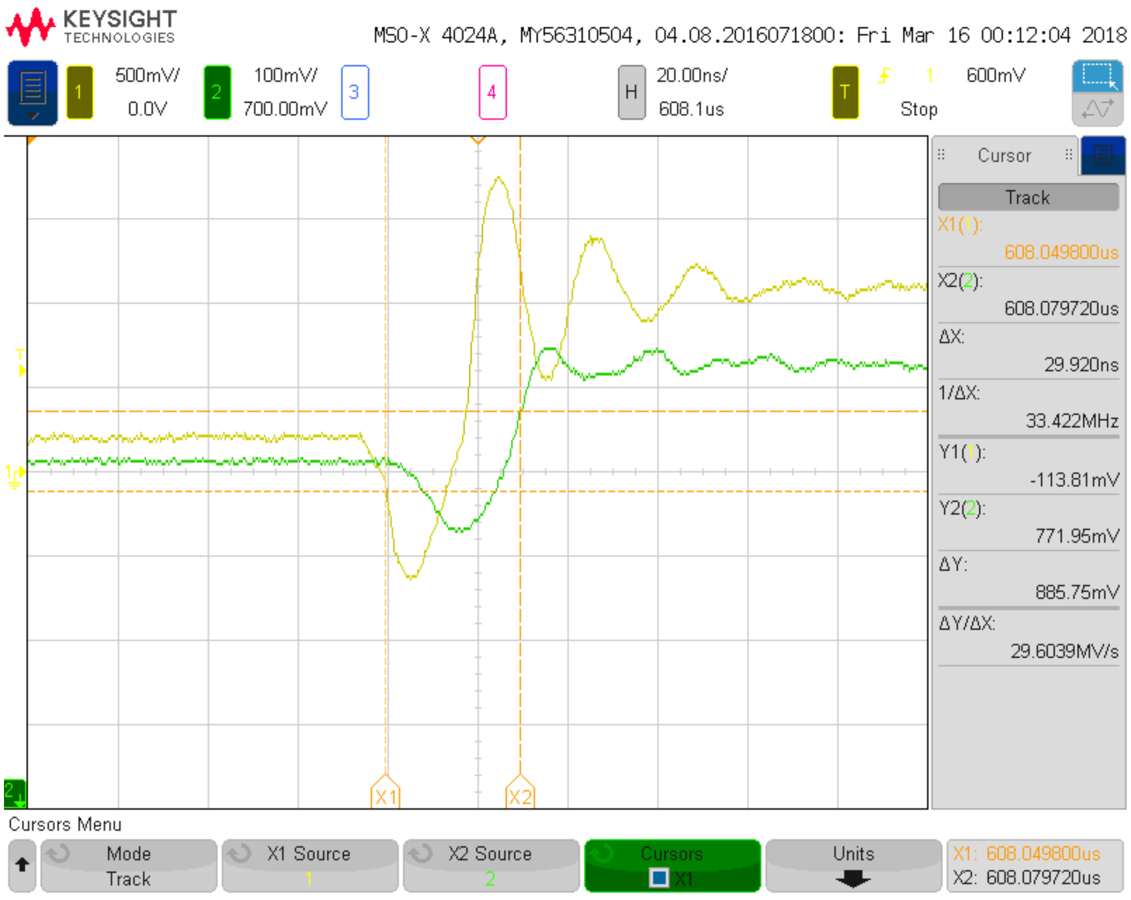
The multiplier's measurement test setup is similar to its simulation testbench. The measurement setup uses the input circuits for test structures section of the PCB to generate voltage input signals. When those signals are applied to the multiplier circuit, they create currents that are similar to those in the simulation testbench. The multiplier takes the input signal from off-chip, weights/scales the signal according to how it is programmed, and then outputs the signal. The output is loaded with the sigmoid test structure for better comparison. Unfortunately, the output is also loaded with the 12 pF oscilloscope probe causing a large RC time constant on the output waveform. This time constant hampers the ability to measure the multiplier at higher frequencies. To work around this issue, the propagation delays at the rising and falling edges are measured.



Freq(1): 4.9782kHz
Pk-Pk(1): 1.1621V
Freq(2): 4.9808kHz
Pk-Pk(2): 464.02mV
Delay(1→2): -92.483us
Delay(1→2): -93.253us

Figure 4.19: Multiplier Measurement Result (Loaded with Sigmoid Input at 5 kHz).

Figure 4.19 depicts the input and output signals for the multiplier testing. The yellow waveform is the input voltage signal used to create the desired currents within the multiplier. The maximum value for this waveform is approximately 1.1 V with the minimum being roughly 200 mV. The frequency for this particular measurement is about 5 kHz. The green waveform is the output voltage waveform and ranges from roughly 520 mV to 720 mV during the switching operation. The multiplier is programmed to the lowest bias settings for this measurement which equates to an approximate doubling of the input signal. Figures 4.20 and 4.21 detail the propagation delays for the rising and falling edges, respectively. The delay for the rising edge is approximately 30 ns while the delay for the falling edge is about 9 ns. These delays are to be expected as the multiplier circuit only has five transistors in the signal pathway and only weights/scales the signal as programmed.

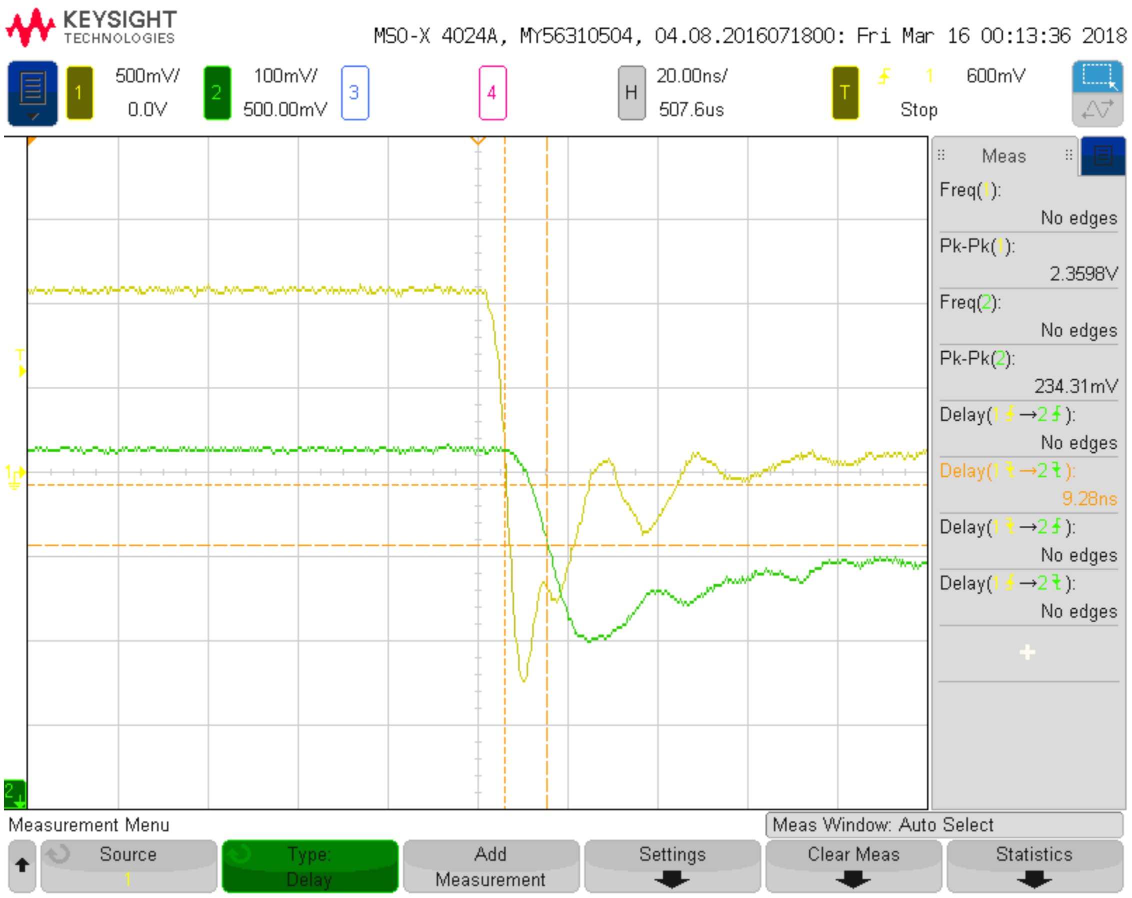

MSO-X 4024A, MY56310504, 04.08.2016071800: Fri Mar 16 00:12:04 2018

1 500mV/ 0.0V
2 100mV/ 700.00mV
H 20.00ns/ 608.1us
T 1 600mV Stop

Cursor
Track
X1(1): 608.049800us
X2(2): 608.079720us
ΔX: 29.920ns
1/ΔX: 33.422MHz
Y1(1): -113.81mV
Y2(2): 771.95mV
ΔY: 885.75mV
ΔY/ΔX: 29.6039MV/s

Cursors Menu
Mode Track
X1 Source 1
X2 Source 2
Cursors X1
Units
X1: 608.049800us
X2: 608.079720us

Figure 4.20: Multiplier Measurement for Rising Edge Propagation Delay.



```
Freq(1):
                 No edges
Pk-Pk(1):
                  2.3598V
Freq(2):
                 No edges
Pk-Pk(2):
                 234.31mV
Delay(1 ↑→2 ↑):
                 No edges
Delay(1 ↓→2 ↓):
                   9.28ns
Delay(1 ↓→2 ↑):
                 No edges
Delay(1 ↑→2 ↓):
                 No edges
```

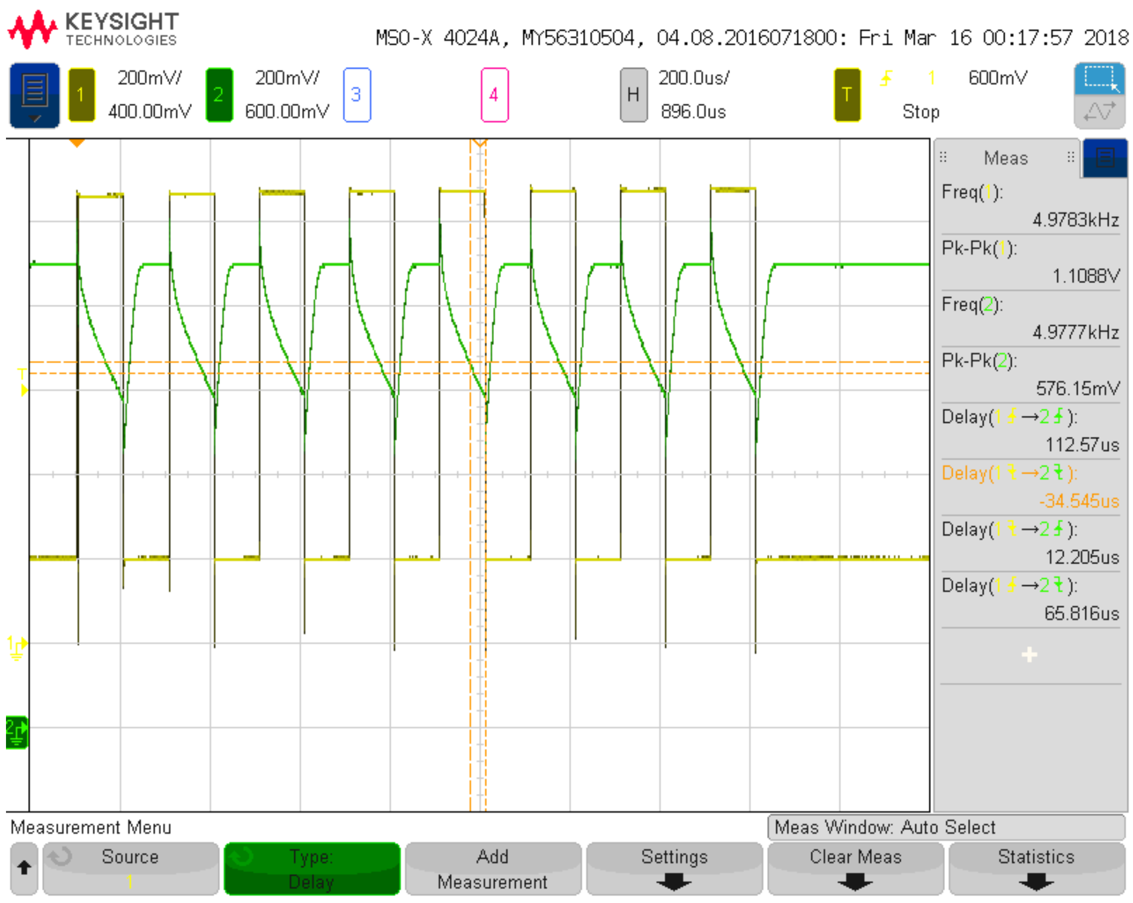
Figure 4.21: Multiplier Measurement for Falling Edge Propagation Delay.

Therefore, the manufactured multiplier circuit behaves in a similar fashion to that in the simulation. Only one programming is used to simplify the results as the multiplier is not the main basis of this discussion. The relatively fast propagation delays provide a bearing that the circuit has the potential of operating at much higher frequencies than desired for this MLP system.

4.2.2 Sigmoid

The measurement setup for the sigmoid design is again similar to its simulation counterpart. Like the multiplier, the test setup for the sigmoid utilizes the input circuits for the test structures section of the PCB to generate the required voltage input signals. The voltage input signal creates a current draw at the input node of the sigmoid circuitry. Once the current is generated by the diode-connected transistor at the input node, it can be transferred to the differential transistor pair at the core of the sigmoid block. After that occurs, the sigmoid behaves in the same fashion as it did previously in the simulation section by applying a logistic function to input signal. The output node is loaded by the multiplier test structure in addition to the 12 pF oscilloscope probe. This loading causes another RC time constant to appear in the output signal of the sigmoid. Like the multiplier, the sigmoid will work around this issue by analyzing the rising and falling edge propagation delays.

Figure 4.22 details the input and output measurements for the sigmoid test structure. The yellow waveform is the input voltage signal to the sigmoid and has the same characteristics as the multiplier measurements (~5 kHz frequency with a max. of 1.1 V and a min. of 200 mV). The green waveform is the output voltage generated as the sigmoid current sources into the multiplier load and charges the capacitive load. The output waveform fluctuates from 760 mV to roughly 1.1 V as the sigmoid moves from an off state to an on state, respectively.



MSO-X 4024A, MY56310504, 04.08.2016071800: Fri Mar 16 00:17:57 2018

1 200mV/ 400.00mV
2 200mV/ 600.00mV
H 200.0us/ 896.0us
T 1 600mV Stop

Meas
Freq(1): 4.9783kHz
Pk-Pk(1): 1.1088V
Freq(2): 4.9777kHz
Pk-Pk(2): 576.15mV
Delay(1 ↑→2 ↑): 112.57us
Delay(1 ↓→2 ↓): -34.545us
Delay(1 ↓→2 ↑): 12.205us
Delay(1 ↑→2 ↓): 65.816us

Figure 4.22: Sigmoid Measurement Result (Loaded with Multiplier at 5 KHz).

The sigmoid is programmed at the lowest bias settings for this measurement result which relates to a reference current of 25 nA and bias current of 200 nA. Figures 4.23 and 4.24 depict the propagation delays for the rising and falling edges, respectively. The rising edge propagation delay is roughly 11 ns while the falling edge propagation delay is about 9 ns for the sigmoid measurement. These delays are again to be expected because of the current mode operation of the sigmoid circuit and the higher current available to charge the load.

In conclusion, the physical sigmoid circuit operates nearly the same as the simulation results. Like the mutliplier results, only one programming is performed to simplify the results as the sigmoid needs only operate as expected for the main MLP system to function properly.

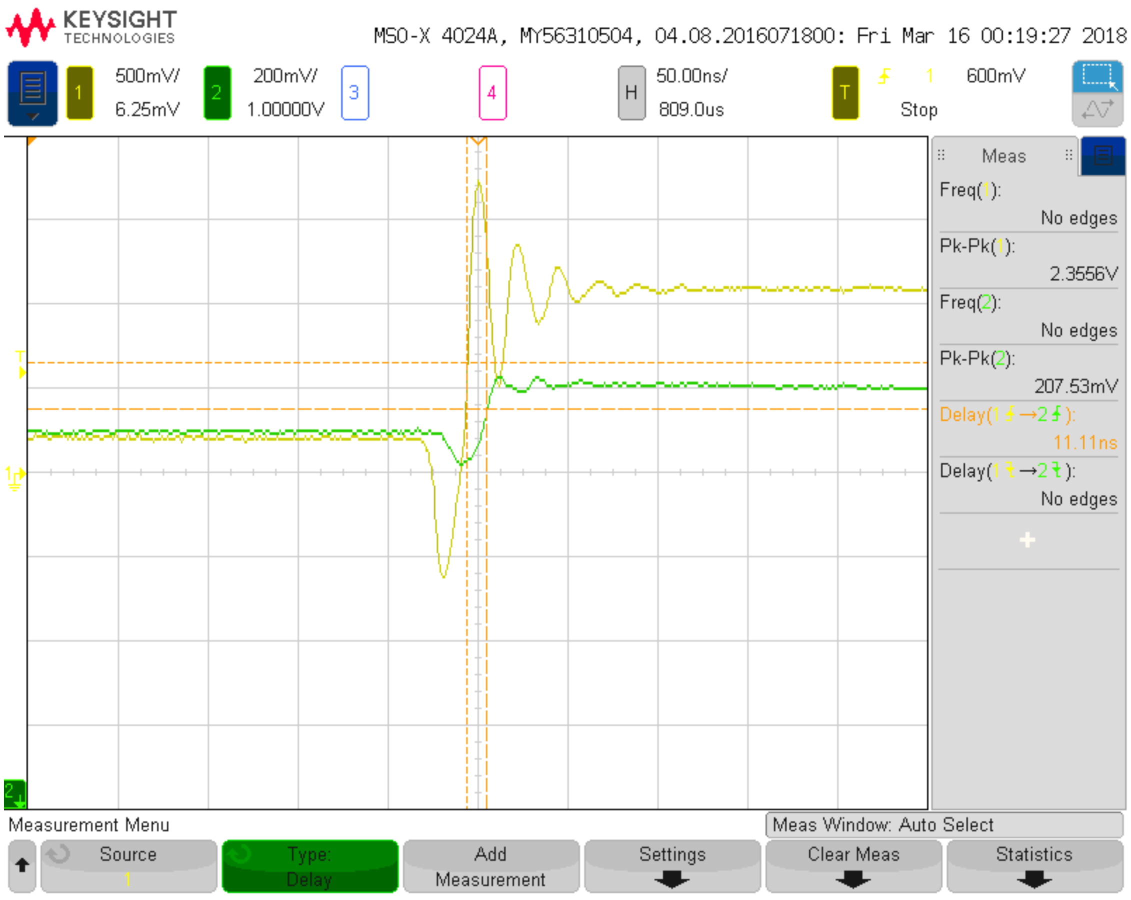
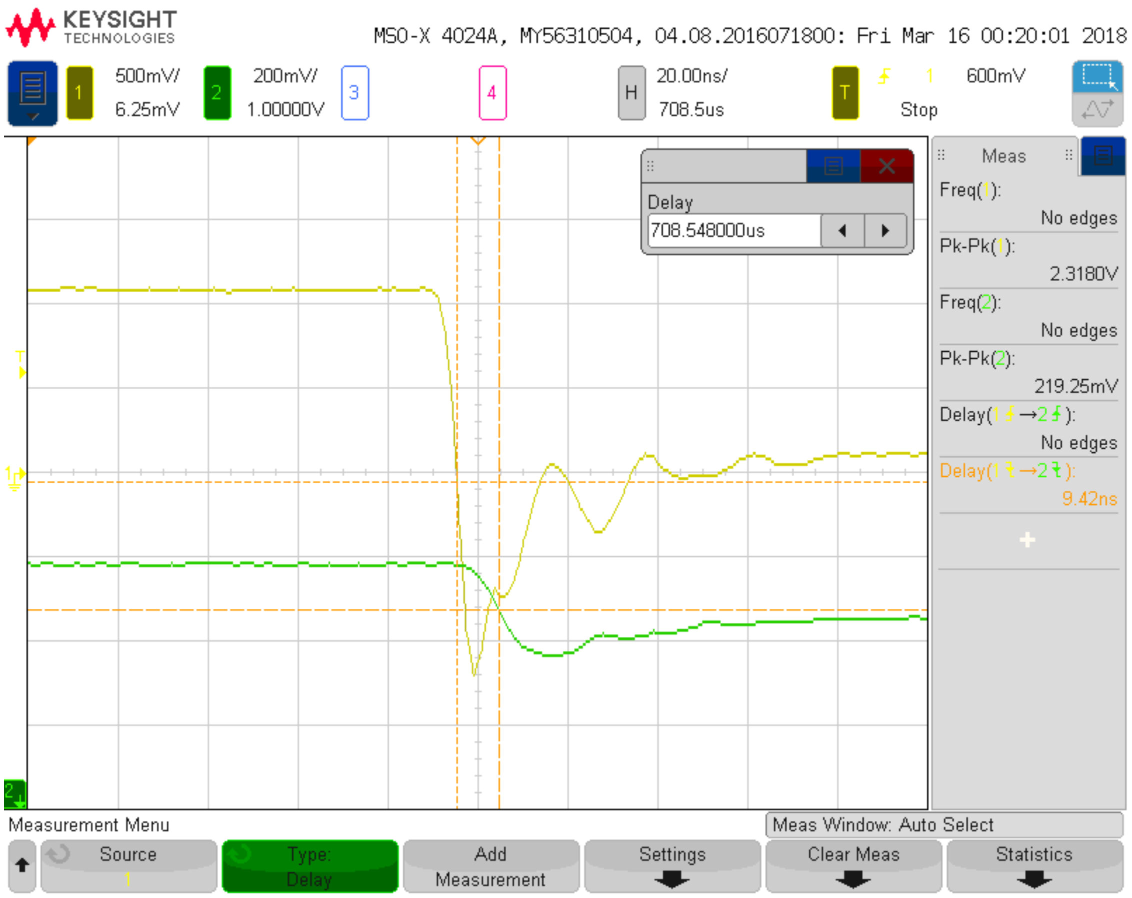


Figure 4.23: Sigmoid Measurement for Rising Edge Propagation Delay.



MSO-X 4024A, MY56310504, 04.08.2016071800: Fri Mar 16 00:20:01 2018

1 500mV/ 6.25mV
2 200mV/ 1.00000V
H 20.00ns/ 708.5us
T 600mV Stop

Delay 708.548000us

Freq(1): No edges
Pk-Pk(1): 2.3180V
Freq(2): No edges
Pk-Pk(2): 219.25mV
Delay(1→2): No edges
Delay(1→2): 9.42ns

Meas Window: Auto Select

Figure 4.24: Sigmoid Measurement for Falling Edge Propagation Delay.

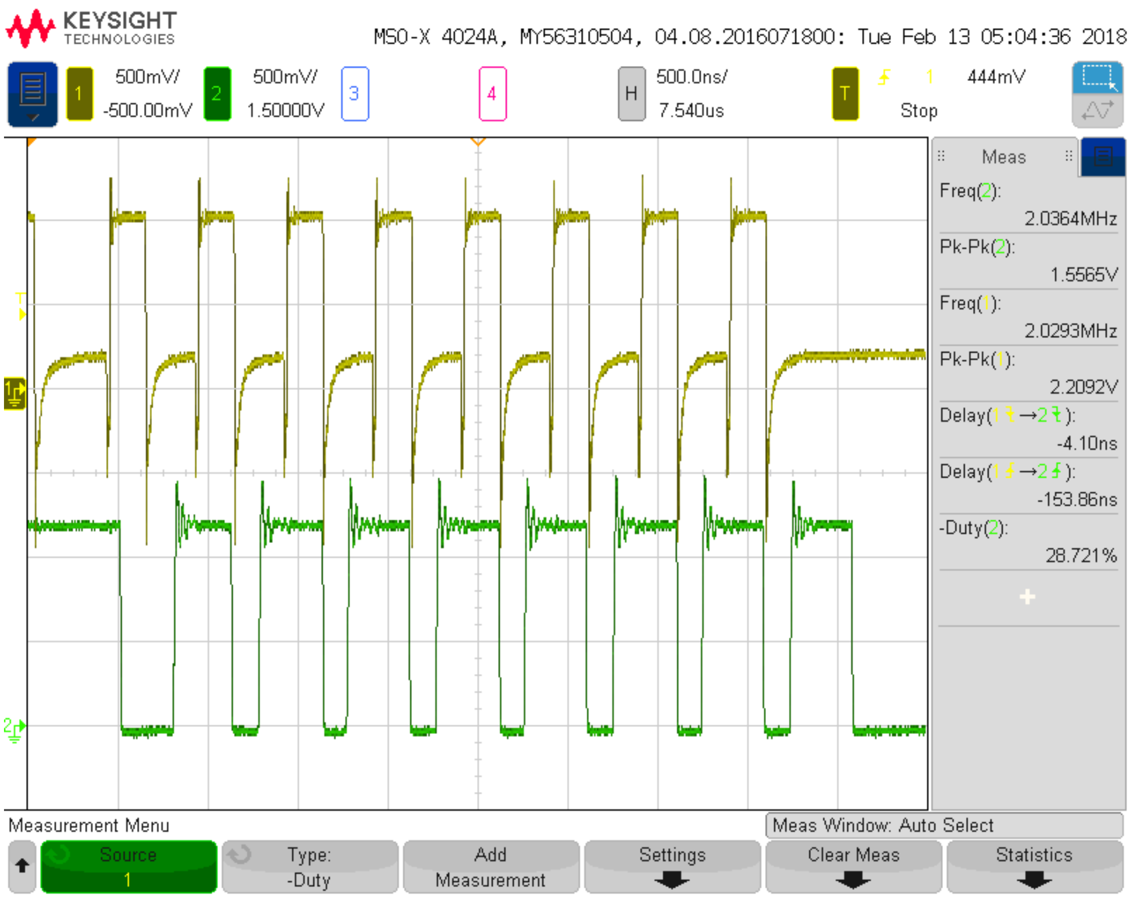
4.2.3 MLP System

The final measurement section discusses the results for several different configurations for the main MLP system as well as a brief noise analysis on the first configuration. The measurement results utilize one or more inputs from the main input signal generating circuit section after the integrated circuit is programmed via the microcontroller. The programming consists of activating switches for the best routing scheme for each configuration and activating the bias cells in order to operate the MLP at the highest possible frequency while still maintaining an accurate output signal. For all of the configurations, the MLP is programmed as a classifier to verify the programming and signal accuracy. Each configuration is judged on its ability to meet the figure of merit of 1 Tera-operations per second per Watt (TOPS/s/W) which is defined by Equation 4.1. Operations consist of either a sum or multiply operation within each activated neuron. In addition to the verfication results for the configuration, the propagation delays for one of the rising and falling edges of the pulse train will be analyzed. All of the configuration will have input voltage signals that have a minimum at 200 mV and a maximum in the range of 900 mV to 1.1 V depending upon the load on the microcontroller's IO ports (lower for higher number of inputs to chip). Lastly, the input signal is operating at the worst case situation which is alternating between high and low states.

$$FOM = \frac{(operations)(frequency)}{Power} \tag{4.1}$$

Figure 4.16 from the simulation section depicts the first configuration analyzed for measurement results. Figure 4.25 details the input voltage waveform in yellow and the output voltage waveform in green for this first configuration. The frequency for the input signal data is 4.06 MHz which is reflected in the output waveform which goes from a low state at zero to a high state at 1.2 V. The measured on state current for this configuration is 14 μA with the off state current being 7μA. These currents averaged over the 8 on states and 8 off states of the signal and multiplied with the voltage give an average power of 12.6 μW. This configuration has 6 total operations (2 per neuron with 3 neurons). Therefore, the FOM is 1.93 TOPS/s/W for this configuration and input signal. Figures 4.26 and 4.27 show the propagation delay for the rising and falling edges, respectively. The rising edge propagation delay is 338 ns while the falling edge is 489 ns. The large delays come from the thresholding circuitry as it requires large currents to change the voltage signal at the comparison node as well as the numerous parasitics encountered from the routing and switches.

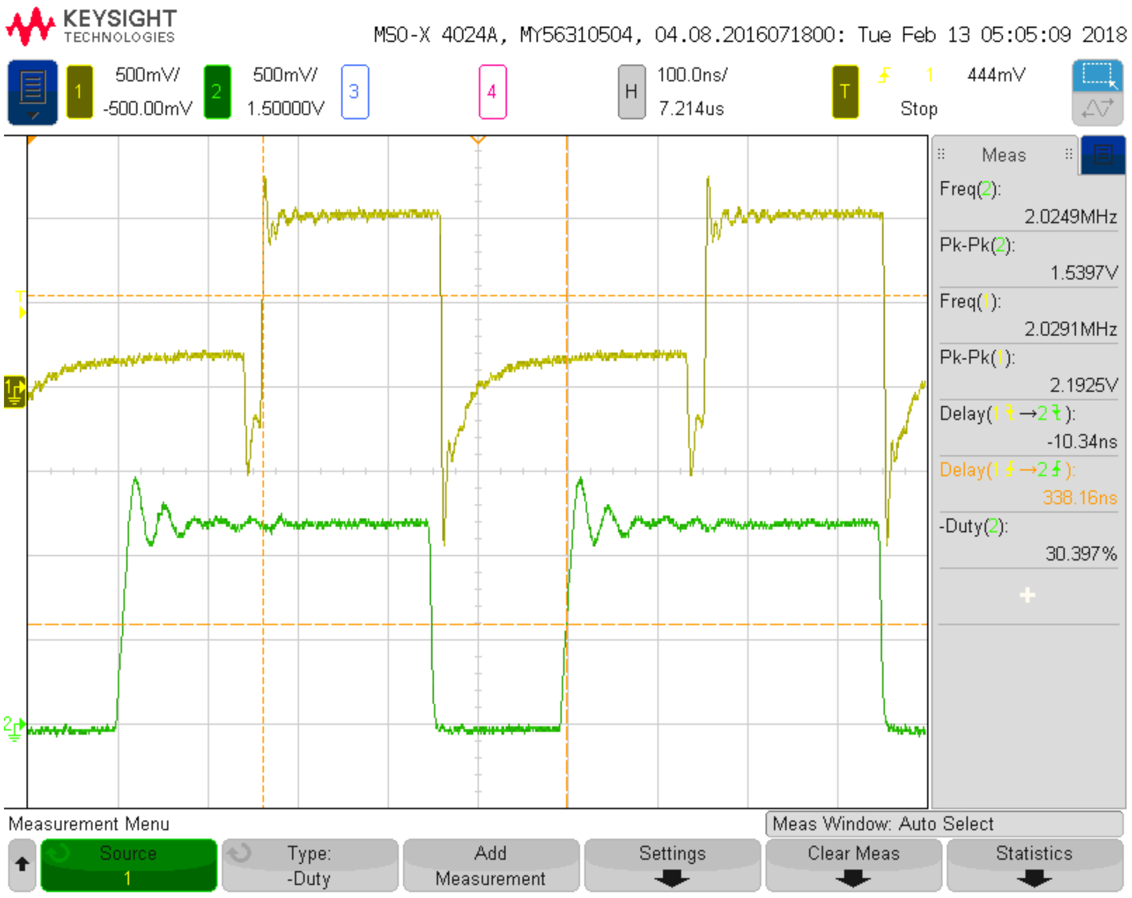
The remainder of the configuration figures are placed in Appendix B as not to overwhelm the result sections with excessive figures. However, their results will be summarized in Table 4.1 showing the important configuration characteristics similar to the discussion in the previous paragraph. All frequencies, currents, power, and FOM in the table are in MHz, μA, μW, and TOPS/s/W respectively. Table 4.2 details each configuration's rising and falling edge propagation delay in ns. After the first configuration, the configuration number will match its corresponding figure captions in the appendix section. Table 4.3 details a comparison of this work against reviewed literature. The table is broken into



```
MSO-X 4024A, MY56310504, 04.08.2016071800: Tue Feb 13 05:04:36 2018
1  500mV/  -500.00mV
2  500mV/  1.50000V
H  500.0ns/  7.540us
T  1  444mV  Stop

Freq(2):
        2.0364MHz
Pk-Pk(2):
        1.5565V
Freq(1):
        2.0293MHz
Pk-Pk(1):
        2.2092V
Delay(1→2):
        -4.10ns
Delay(1→2):
        -153.86ns
-Duty(2):
        28.721%
```

Figure 4.25: First MLP System Configuration Measurement (No Load at 4.07 MHz).

several comparing features of the different MLP structures with the units for each feature being: for the computation rate in Mega-connections per second (MCPS), power in micro-Watts (μW), synapses (number of multipliers), rate per synapse (MCPS), power per synapse (μW), and FOM (TOPS/s/W). The biggest takeaways from Table 4.3 are that the power per synapse is the lowest for this work as well as achieving the highest FOM compared to the prior art. While the computation rate per synapse is not the best, this can be improved upon by either scaling up the number of available synapses and/or increasing the overall data rate of the system.


KEYSIGHT TECHNOLOGIES

MSO-X 4024A, MY56310504, 04.08.2016071800: Tue Feb 13 05:05:09 2018

1 500mV/ -500.00mV

2 500mV/ 1.50000V

H 100.0ns/ 7.214us

T 1 444mV Stop

Meas

Freq(2): 2.0249MHz

Pk-Pk(2): 1.5397V

Freq(1): 2.0291MHz

Pk-Pk(1): 2.1925V

Delay(1→2): -10.34ns

Delay(1→2): 338.16ns

-Duty(2): 30.397%

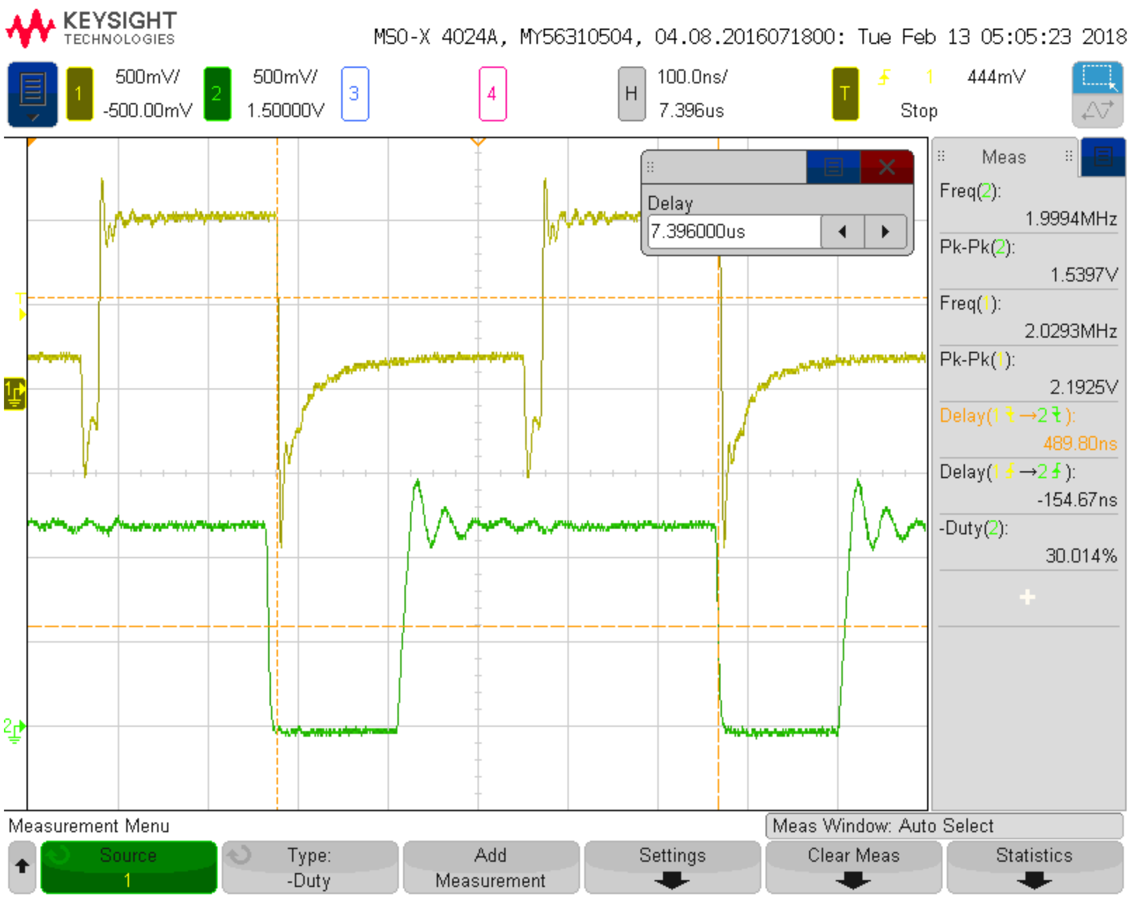
Measurement Menu

Meas Window: Auto Select

Figure 4.26: First MLP System Configuration Measurement for Rising Edge Propagation Delay.

Table 4.1: Summary of Measurement Results for Different MLP System Configurations (see text for units).

Config #	Inputs	Neurons	Freq.	On Current	Off Current	Power	FOM
1	1	3	4.06	14	7	12.6	1.93
2	1	4	4.06	14.1	7.3	12.84	2.52
3	1	5	4.06	19.4	8.6	16.8	2.90
4	1	7	4.06	18.9	6.3	15.12	5.91
5	1	9	2.75	22.8	7.4	18.12	4.56
6	1	12	2.75	26.6	15.5	25.26	5.23
7	4	12	1.51	35.8	17	31.68	2.86
8	1	6	2.04	13.5	4	10.5	2.09
9	2	6	2.57	19	8.1	16.26	2.85
10	1	7	4.06	21.6	8.2	17.88	4.54


MSO-X 4024A, MY56310504, 04.08.2016071800: Tue Feb 13 05:05:23 2018

1 500mV/ -500.00mV
2 500mV/ 1.50000V
H 100.0ns/ 7.396us
T 1 444mV Stop

Delay
7.396000us

Meas
Freq(2): 1.9994MHz
Pk-Pk(2): 1.5397V
Freq(1): 2.0293MHz
Pk-Pk(1): 2.1925V
Delay(1→2): 489.80ns
Delay(1→2): -154.67ns
-Duty(2): 30.014%

Meas Window: Auto Select

Figure 4.27: First MLP System Configuration Measurement for Falling Edge Propagation Delay.

Table 4.2: Summary of Propagation Delays for Different MLP System Configurations (see text for units).

Config #	Inputs	Neurons	Freq.	Delay (Rising)	Delay (Falling)
1	1	3	4.06	338	489
2	1	4	4.06	345	457
3	1	5	4.06	343	479
4	1	7	4.06	436	492
5	1	9	2.75	515	585
6	1	12	2.75	541	621
7	4	12	1.51	555	889
8	1	6	2.04	829	750
9	2	6	2.57	610	687
10	1	7	4.06	395	564

Table 4.3: MLP Comparison with Prior Art (see text for units).

	Comp. Rate	Power	Syn. #	Rate per Syn.	Power per Syn.	FOM
Diotalevi *et al.*	30.00	3.00	1	30.00	3.00	-
Gatet *et al.*	2400	595000	10	240	59500	0.004
Bo *et al.*	1000	25000	28	35.70	892.90	0.040
Bo *et al.*	2500	200000	4810	0.52	41.60	0.012
Park *et al.*	411300	213100	2056	200	103.65	1.930
Tsai *et al.*	860160	310000	4096	210	75.68	1.450
This work	86.50	31.68	48	1.80	0.66	**2.856**

The noise analysis for the configuration in Figure 4.16 is the last measurement performed on the main MLP system. The noise is measured with the MLP set up as a simple classifier with two classes (OFF being class 0 and ON being class 1). An input is connected to a current source meter in order to provide the input current to measure input referred noise. When the input is near the decision boundary for the MLP and the classification operation is performed, the noise will cause the output to become uncertain. Assuming additive Guassian noise, the relative frequency of the class 1 output will be shown to approach the cumulative density function (CDF) of the normal distribution. The standard deviation σ of this distribution can be extracted from the data and can be analyzed as the input referred rms noise of the system. Two noise analysis runs are performed on this configuration and are shown in Figures 4.28 and 4.29. The measured input referred noise for the first run is 435.9 pA_{rms} with the second run giving a similar result of 432.13 pA_{rms}. With a full-scale input of 100 nA (or greater), the SNRs of the MLP system for this configuration for the first and second run are 47.21 dB and 47.29 dB, respectively.

In conclusion, the MLP system has operated consistently and above the desired figure of merit for the system. The configurablility of the system has been shown successfully through the ten sample configurations analyzed. Additionally, the low power aspects of the MLP system are prevalent in the FOMs above 1 TOPS/s/W. These higher FOM numbers could be sacrificed slightly to improve the propagation delays of the rising and falling edges as well as push the frequency of the system while still maintaining above 1 TOPS/s/W.

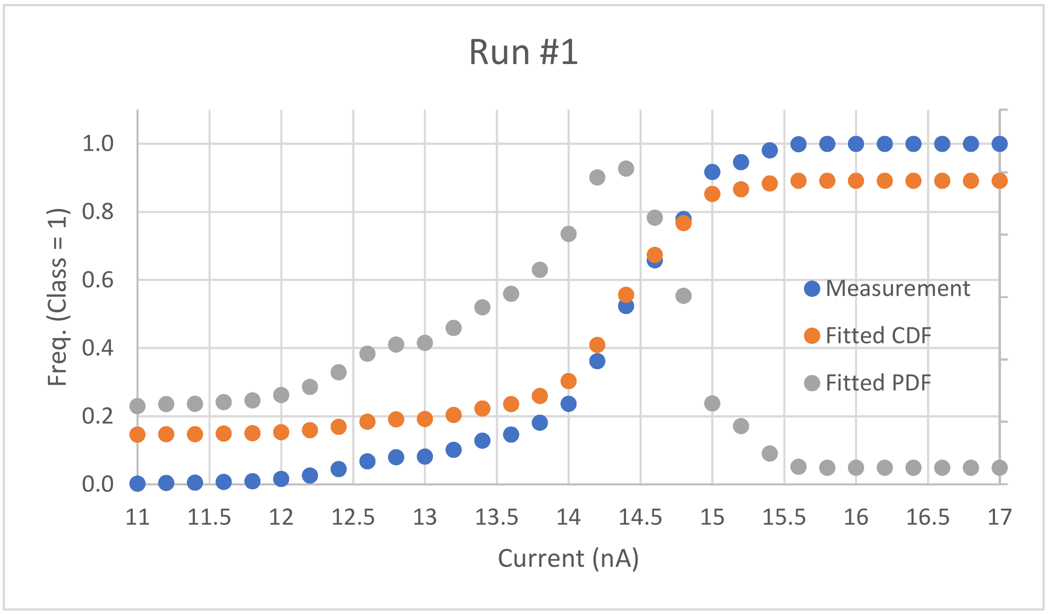


Figure 4.28: First Noise Analysis Run on MLP System.

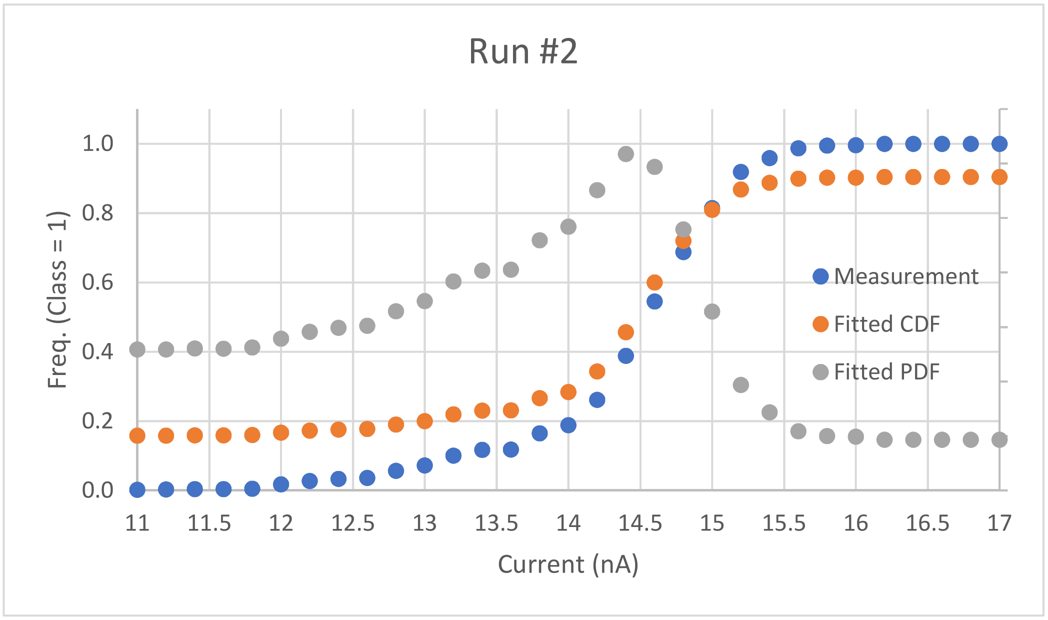


Figure 4.29: Second Noise Analysis Run on MLP System.

Chapter 5

Conclusions and Future Work

5.1 Conclusions and Original Contributions

The MLP system design is centered around the ability to be both a configurable and low power analog network as analog multipliers and sigmoid circuits are already proven concepts, which is shown in Chapter 2. The first criterion of configurability is successfully demonstrated by the MLP system in Chapter 4 and Appendix B via the ten configuration samples that produce the correct, expected outputs. The low power criterion is shown through the application and calculation of a figure of merit that is greater than 1 Tera-operations per second per Watt in each configuration. Unfortunately, the full potential of the MLP system could not be tested because of the limitations, discussed in Chapter 4, that hampered the input signal frequency range. However even with these limitations on the data rate of the system, the MLP is still capable of operating at the original FOM which is intended for frequencies in the range of 100's of MHz. Therefore, the MLP system should easily meet its original goals should it be redesigned. The system programming allows for a wide range of applications from image analysis to signal processing to pattern recognition. The MLP structure is well suited for this diverse range of applications if it is scaled up to become more of a true neural network as right now it is limited by connections and the number of inputs it can sustain.

The amount that the MLP system would have to be scaled up depends upon the nature of the intended application. However for typical neural network applications, this MLP system would have to be improved to have at least 100 neurons if not 100's to behave similarly to that of other modern neural networks. In order to achieve this highly scaled up version, the integrated circuit would have to be much larger and more compact as the physical layout is not 100% optimized for saving space. The integrated circuit would eventually be limited by the number of interconnect layers in the CMOS manufacturing technology being used. However, the MLP system as of right now could be highly desirable for small-scale embedded applications and biomedical applications. The rise of the Internet of Things has given way for products to be more interconnected and "intelligent" as products move away from direct consumer control. With that in mind, the MLP system could be implanted in home consumer products such as thermostats, lights, refridgerators, and other products to take in sensor data and output classified signals to a central hub that utilizes the data and regulates the consumer's home with minimal human interaction. For biodmedical applications, the MLP could be paired with sensors that are either implanted or placed onto a patient to monitor and output a signal when the patient experiences certain medical criteria (such as breathing events in sleep apnea patients). The obstacles that stand in between the MLP and placement in finalized products are the following: the input signal structures are burdensome and need refining on-chip or otherwise, the MLP requires reprogramming with every loss of power event, and the architecture needs optimization to improve overall functionality.

The original contributions of this work include several of the features touched upon in the first paragraph. The first contribution is high speed and small form factor multiplier and sigmoid circuits that are capable of operating at a frequency of at least 100 MHz which is proven via the propagation delays for both circuits. Additionally, both circuits are designed to utilize as little physical layout space on the integrated circuit as possible for the topologies and fabrication technology chosen. The next contribution comes from the configurable nature of the MLP system. The MLP has proven to be easily altered to take whichever desired neural network shape is possible with its available resources. The biasing programming also allows the input signals to be controlled however the user dictates. The third contribution is the low power analog design of the entire system which grants the possibility of a mobile MLP integrated circuit that consumes little power and has a high throughput. The last contribution is a scalable system that grants the ability to easily scale up or down the size of the overall system to create enough available resources to perform any MLP neural network task.

5.2 Future Work

This final section discusses the changes that could be made to the MLP network in order to improve the system functionality. The first change would be a faster and more efficient current comparison network in the thresholding circuit cell. The TC cell bogs down the system's data rate by providing the largest addition to the propagation delay as well as consuming the most current in the system. The second change would be to create input structures on-chip to convert the voltage input signals from the microcontroller to current signals more readily and with less parasitics than the support circuitry off-chip. The bias current mirrors that relay the master bias current throughout the chip could be improved upon in order to have a more accurate bias current that is similar in all sections of the chip. The output inverters on the OR gates or the TC cells should be expanded to have a greater current drive capability (such as exponential horn technique [43]) in order to drive at least the capacitive load of an oscilloscope probe. Another change could be better routing structures so as to reduce the parasitics seen in the signal pathways and decrease propagation delay times. Also, the test board parasitics should be analyzed and limited as best as possible to increase the maximum data rate potential. The final and most important change would be to choose a higher frequency capable microcontroller unit in order to successfully operate the MLP system in the 100's of MHz frequency range. In conclusion, these changes would allow the MLP to operate at its maximum limits potentially achieving a greater figure of merit.

Bibliography

[1] M. Gales, "Module 4f10: Statistical Pattern Processing Handout 8: Multi-Layer Perceptrons," Faculty of Engineering, University of Cambridge, Cambridge, UK, 2015. x, 1, 2, 3, 4, 20

[2] D. A. Pomerleau, "Alvinn, an autonomous land vehicle in a neural network," Carnegie Mellon University, Computer Science Department, Tech. Rep., 1989. x, 9, 10

[3] J. B. Lont and W. Guggenbuhl, "Analog cmos implementation of a multilayer perceptron with nonlinear synapses," *IEEE Transactions on Neural Networks*, vol. 3, no. 3, pp. 457–465, May 1992. x, 6, 11

[4] L. Gatet, H. Tap-Beteille, and M. Lescure, "Analog neural network design for real-time surface detection with a laser rangefinder," in *2007 IEEE Instrumentation Measurement Technology Conference IMTC 2007*, May 2007, pp. 1–6. x, 11, 12, 20

[5] F. Diotalevi, M. Valle, G. Bo, E. Biglieri, and D. Caviglia, "Analog cmos current mode neural primitives," in *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, vol. 2. IEEE, 2000, pp. 717–720. x, 13, 20

[6] D. Maliuk and Y. Makris, "A dual-mode weight storage analog neural network platform for on-chip applications," in *2012 IEEE International Symposium on Circuits and Systems*, May 2012, pp. 2889–2892. x, 14, 15, 30, 32, 41

[7] G. Bo, D. Caviglia, and M. Valle, "A current mode cmos multi-layer perceptron chip," in *Microelectronics for Neural Networks, 1996., Proceedings of Fifth International Conference on*. IEEE, 1996, pp. 103–106. x, 14, 15, 20

[8] R. M. d. Silva, L. d. M. Mourelle, and N. Nedjah, "Compact yet efficient hardware architecture for multilayer-perceptron neural networks," *Sba: Controle & Automação Sociedade Brasileira de Automatica*, vol. 22, no. 6, pp. 647–663, 2011. x, 16, 17

[9] Z. Su, B. M. Wilamowski, R. Wang, and F. F. Dai, "Adaptive integratable hardware realization of analog neural networks for nonlinear system," in *2015 IEEE 13th*

International Conference on Industrial Informatics (INDIN), July 2015, pp. 521–526. x, 16, 19

[10] B. Gilbert, “Translinear circuits: An historical overview,” *Analog Integrated Circuits and Signal Processing*, vol. 9, no. 2, pp. 95–118, 1996. x, 21, 22

[11] D. Coue and G. Wilson, “A four-quadrant subthreshold mode multiplier for analog neural-network applications,” *IEEE Transactions on Neural Networks*, vol. 7, no. 5, pp. 1212–1219, Sep 1996. x, 22, 23

[12] B. M. Wilamowski, “Vlsi analog multiplier/divider circuit,” in *Industrial Electronics, 1998. Proceedings. ISIE ’98. IEEE International Symposium on*, vol. 2, Jul 1998, pp. 493–496 vol.2. x, 23, 24

[13] R. B. dos Santos, P. M. S. R. Rizol, and L. Mesquita, “Design of cmos current-mode multiplier-divider circuit for type-2 flc applications,” in *2015 IEEE 6th Latin American Symposium on Circuits Systems (LASCAS)*, Feb 2015, pp. 1–4. x, 25, 26

[14] A. Baharmast, S. J. Azhari, and S. Mowlavi, “A new current mode high speed four quadrant cmos analog multiplier,” in *2016 24th Iranian Conference on Electrical Engineering (ICEE)*, May 2016, pp. 1371–1376. x, 25, 27

[15] A. Mahmoudi, A. Khoei, and K. Hadidi, “A novel current-mode micropower four quadrant cmos analog multiplier/divider,” in *2007 IEEE Conference on Electron Devices and Solid-State Circuits*, Dec 2007, pp. 321–324. x, 26, 28

[16] M. Valle and F. Diotalevi, “An analog cmos four quadrant current-mode multiplier for low power artificial neural networks implementation,” *Dep. of Biophysical and Electronic Engineering, Univ. of Genova*, 2001. x, 27, 29

[17] ——, “A dedicated very low power analog vlsi architecture for smart adaptive systems,” *Applied Soft Computing*, vol. 4, no. 3, pp. 206–226, 2004. x, 31, 33

[18] J. Lazzaro, S. Ryckebusch, M. Mahowald, and C. Mead, “Winner-take-all networks of o (n) complexity,” in *NIPS*, vol. 1, 1988, pp. 703–711. x, 33, 34

[19] H. Y. Hsieh, K. T. Tang, Z. H. Tsai, and H. Chen, "A low-power, high-resolution wta utilizing translinear-loop pre-amplifier," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, July 2010, pp. 1–5. xi, 35

[20] A. Fish, V. Milrud, and O. Yadid-Pecht, "High-speed and high-precision current winner-take-all circuit," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 52, no. 3, pp. 131–135, March 2005. xi, 36, 37

[21] M. Padash, A. Khoei, K. Hadidi, and H. Ghasemiyan, "A high precision high frequency vlsi multi-input min-max circuit based on wta-lta cells," in *2011 International Conference on Electronic Devices, Systems and Applications (ICEDSA)*, April 2011, pp. 152–156. xi, 37, 38

[22] K. Wawryn and B. Strzeszewski, "Prototype current mode circuits for programmable wta network," in *Electronics, Circuits and Systems, 1999. Proceedings of ICECS '99. The 6th IEEE International Conference on*, vol. 2, Sep 1999, pp. 1013–1016 vol.2. xi, 38, 39

[23] H. Yu, R. S. Miyaaoka, and T. K. Lewellen, "A high-speed and high-precision winner-select-output (wso) asic," in *1997 IEEE Nuclear Science Symposium Conference Record*, Nov 1997, pp. 656–660 vol.1. xi, 40

[24] J. Lu, S. Young, I. Arel, and J. Holleman, "A 1 tops/w analog deep machine-learning engine with floating-gate storage in 0.13 μm cmos," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 270–281, Jan 2015. xi, 41, 42

[25] R. B. Wunderlich, F. Adil, and P. Hasler, "Floating gate-based field programmable mixed-signal array," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 8, pp. 1496–1505, Aug 2013. xi, 42, 43, 56

[26] D. MacKay, *Information Theory, Inference, and Learning Algorithms.* Cambridge, UK: Cambridge University Press, 2003. 1

[27] P. Hasler, "Low-power programmable signal processing," in *Fifth International Workshop on System-on-Chip for Real-Time Applications (IWSOC'05)*, July 2005, pp. 413–418. 5

[28] M. Gravati, M. Valle, G. Ferri, N. Guerrini, and N. Reyes, "A novel current-mode very low power analog cmos four quadrant multiplier," in *Proceedings of the 31st European Solid-State Circuits Conference, 2005. ESSCIRC 2005.*, Sept 2005, pp. 495–498. 6

[29] M. A. Al-Absi, A. Hussein, and M. T. Abuelma'atti, "A novel current-mode ultra low power analog cmos four quadrant multiplier," in *2012 International Conference on Computer and Communication Engineering (ICCCE)*, July 2012, pp. 13–17. 6

[30] T. Talaka, M. Kolasa, R. Dugosz, and W. Pedrycz, "Analog programmable distance calculation circuit for winner takes all neural network realized in the cmos technology," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 3, pp. 661–673, March 2016. 6

[31] M. Riedmiller, "Machine Learning: Multi Layer Perceptrons," Faculty of Engineering, Albert-Ludwigs-University, Freiburg, DE, 2009. 6

[32] J. A. Bullinaria, "Applications of Multi-Layer Perceptrons," Faculty of Computer Science, The University of Birmingham, Birmingham, UK, 2015. 9

[33] L. Gatet, H. Tap-Beteille, M. Lescure, D. Roviras, and A. Mallet, "Design and test of a cmos mlp analog neural network for fast on-board signal processing," in *2006 13th IEEE International Conference on Electronics, Circuits and Systems*, Dec 2006, pp. 922–925. 14, 20

[34] G. M. Bo, D. D. Caviglia, H. Chible, and M. Valle, "Design of an analog cmos self-learning mlp chip," in *Circuits and Systems, 1998. ISCAS '98. Proceedings of the 1998 IEEE International Symposium on*, vol. 3, May 1998, pp. 46–49 vol.3. 14, 20

[35] G. A. Cairns and L. Tarassenko, "Implementation issues for on-chip learning with analogue vlsi mlps," in *1995 Fourth International Conference on Artificial Neural Networks*, Jun 1995, pp. 465–470. 16

[36] T. Talaka, M. Kolasa, R. Dugosz, and W. Pedrycz, "Analog programmable distance calculation circuit for winner takes all neural network realized in the cmos technology," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 3, pp. 661–673, March 2016. 19

[37] S. W. Park, J. Park, K. Bong, D. Shin, J. Lee, S. Choi, and H. J. Yoo, "An energy-efficient and scalable deep learning/inference processor with tetra-parallel mimd architecture for big data applications," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 9, no. 6, pp. 838–848, Dec 2015. 19

[38] C. H. Tsai, W. J. Yu, W. H. Wong, and C. Y. Lee, "A 41.3/26.7 pj per neuron weight rbm processor supporting on-chip learning/inference for iot applications," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 10, pp. 2601–2612, Oct 2017. 19

[39] B. Gilbert, "Translinear circuits: A proposed classification," *Electronics letters*, vol. 11, no. 1, pp. 14–16, 1975. 21

[40] A. J. Lopez-Martin and A. Carlosena, "Design of mos-translinear multiplier/dividers in analog vlsi," *VLSI Design*, vol. 11, no. 4, pp. 321–329, 2000. 21

[41] C. Mead, *Analog VLSI and Neural Systems*, ser. Addison-Wesley VLSI system series. Addison-Wesley, 1989. [Online]. Available: https://books.google.com/books?id=-j8PAQAAMAAJ 29, 30

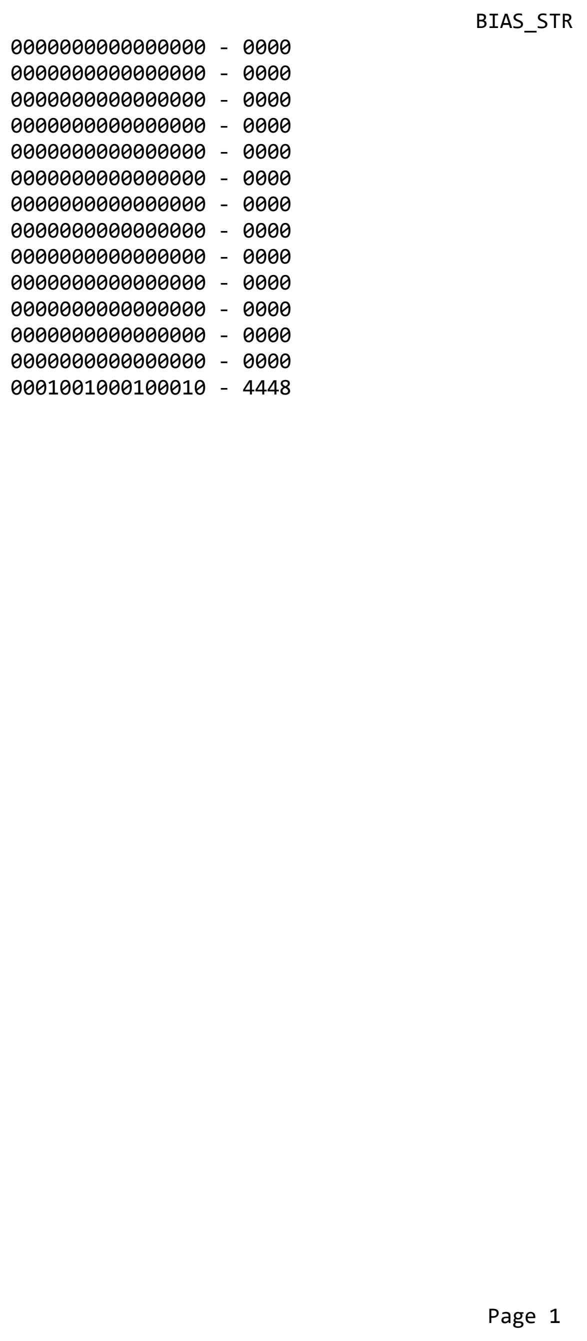
[42] B. A. Minch, "A low-voltage mos cascode current mirror for all current levels," in *The 2002 45th Midwest Symposium on Circuits and Systems, 2002. MWSCAS-2002.*, vol. 2, Aug 2002, pp. II–53–6 vol.2. 46

[43] T. Maekawa, S. Amakawa, N. Ishihara, and K. Masu, "Design of cmos inverter-based output buffers adapting the cherry-hooper broadbanding technique," in *2009 European Conference on Circuit Theory and Design*, Aug 2009, pp. 511–514. 98

Appendix

Appendix A

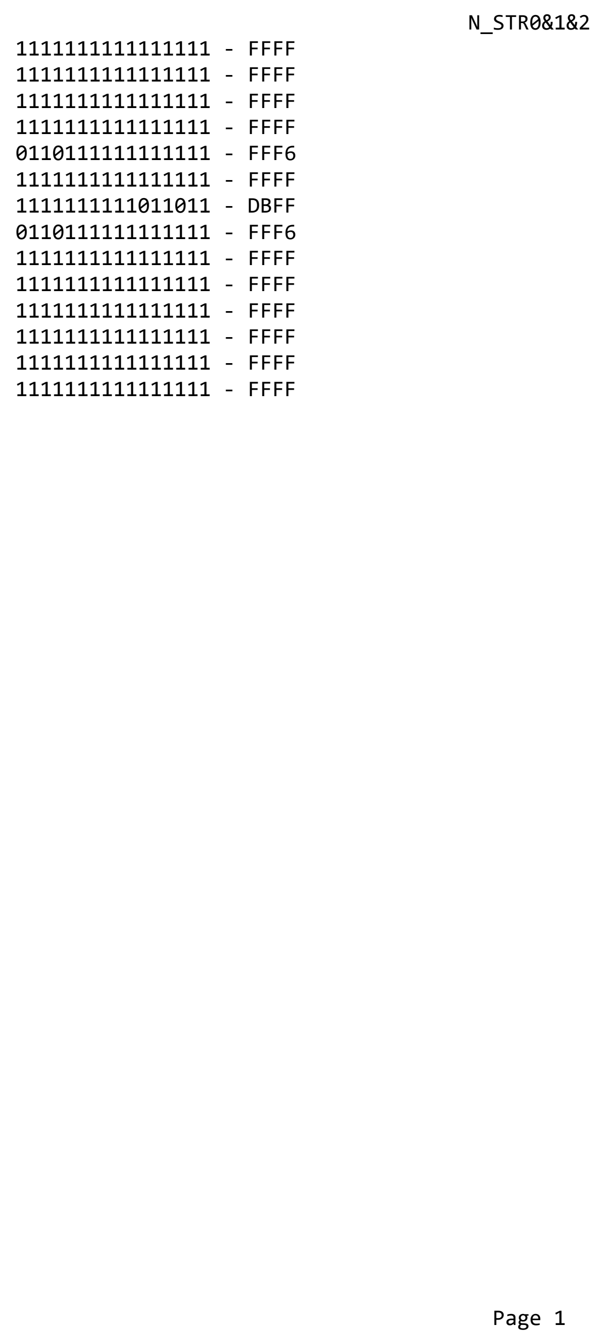
Code for Programming MLP System



```
                          BIAS_STR
0000000000000000 - 0000
0000000000000000 - 0000
0000000000000000 - 0000
0000000000000000 - 0000
0000000000000000 - 0000
0000000000000000 - 0000
0000000000000000 - 0000
0000000000000000 - 0000
0000000000000000 - 0000
0000000000000000 - 0000
0000000000000000 - 0000
0000000000000000 - 0000
0000000000000000 - 0000
0001001000100010 - 4448
```

Page 1

Figure A.1: Simulation Code for Bias Bitstream.

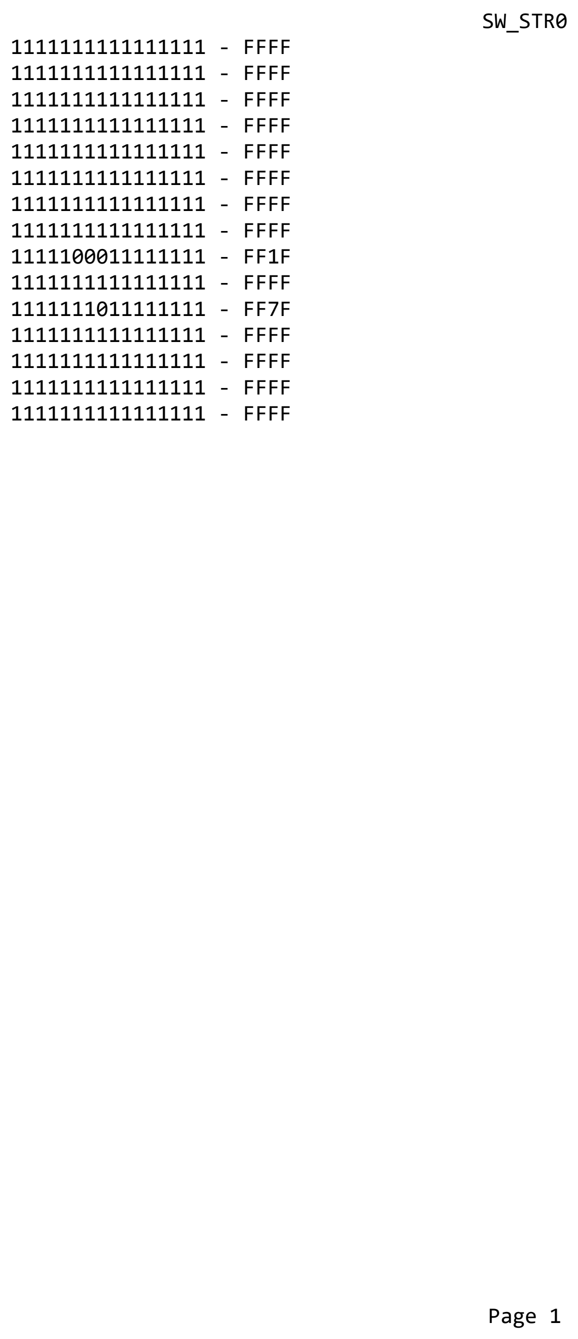


N_STR0&1&2

```
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
0110111111111111 - FFF6
1111111111111111 - FFFF
1111111111011011 - DBFF
0110111111111111 - FFF6
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
```

Page 1

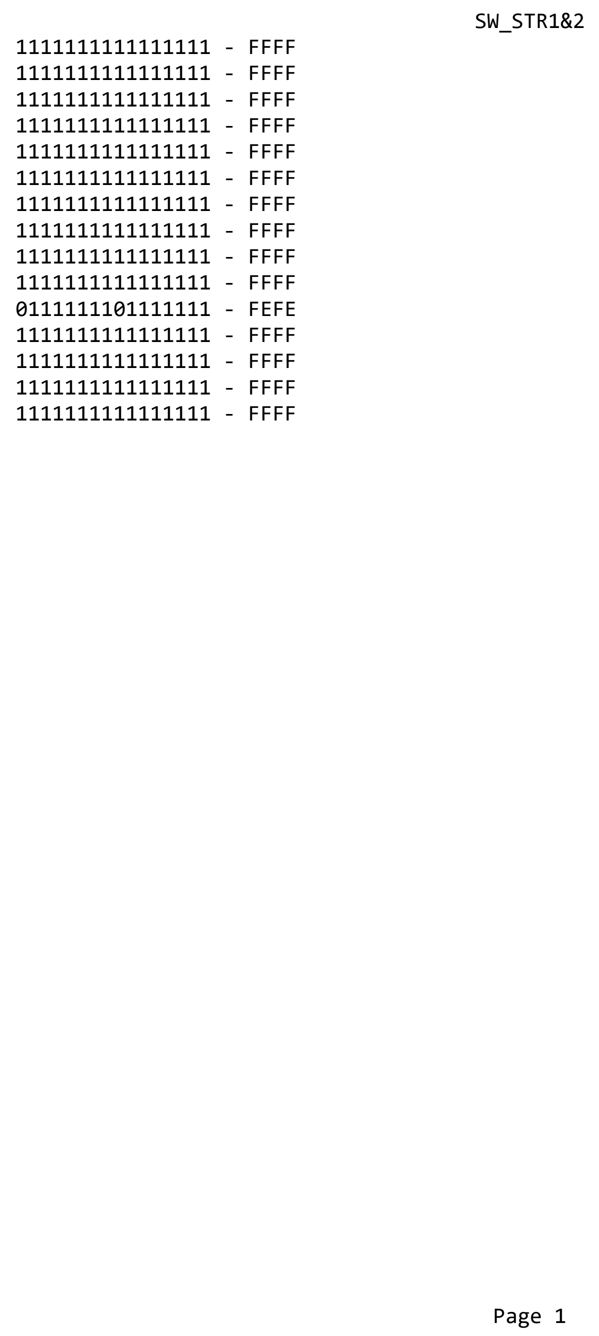
Figure A.2: Simulation Code for All Neuron Bitstreams.



```
                                  SW_STR0
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111100011111111 - FF1F
1111111111111111 - FFFF
1111111011111111 - FF7F
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
```

Page 1

Figure A.3: Simulation Code for the First Switch Bitstream.

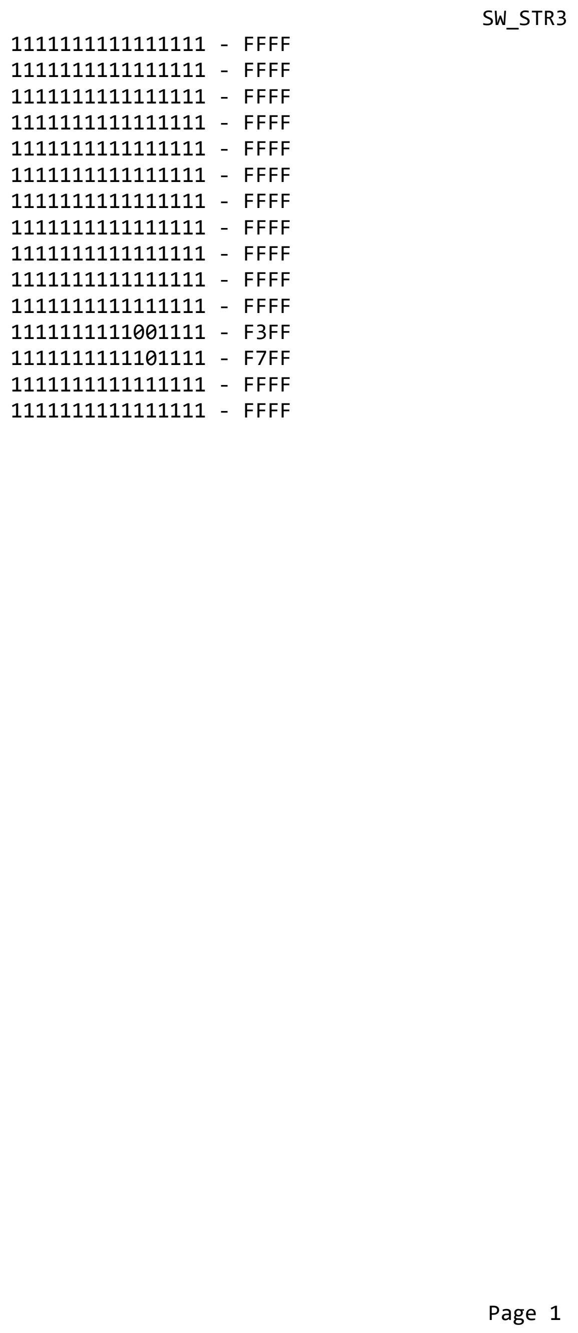


SW_STR1&2

```
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
0111111101111111 - FEFE
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
```

Page 1

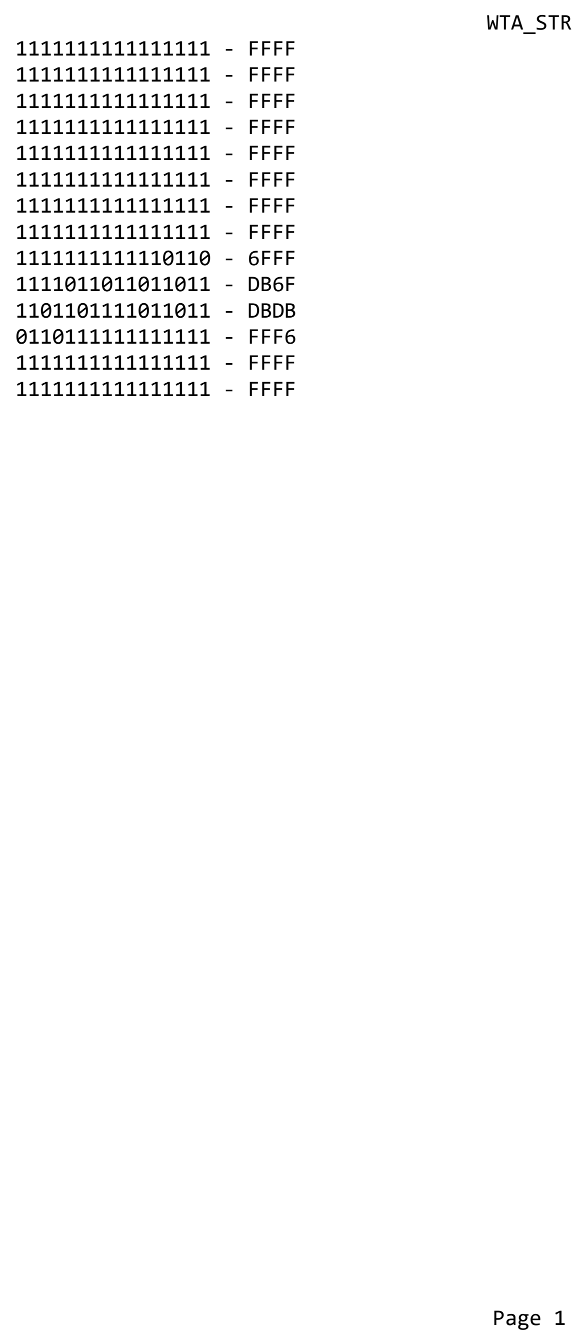
Figure A.4: Simulation Code for Second and Third Switch Bitstreams.



```
                              SW_STR3
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111001111 - F3FF
1111111111101111 - F7FF
1111111111111111 - FFFF
1111111111111111 - FFFF
```

Page 1

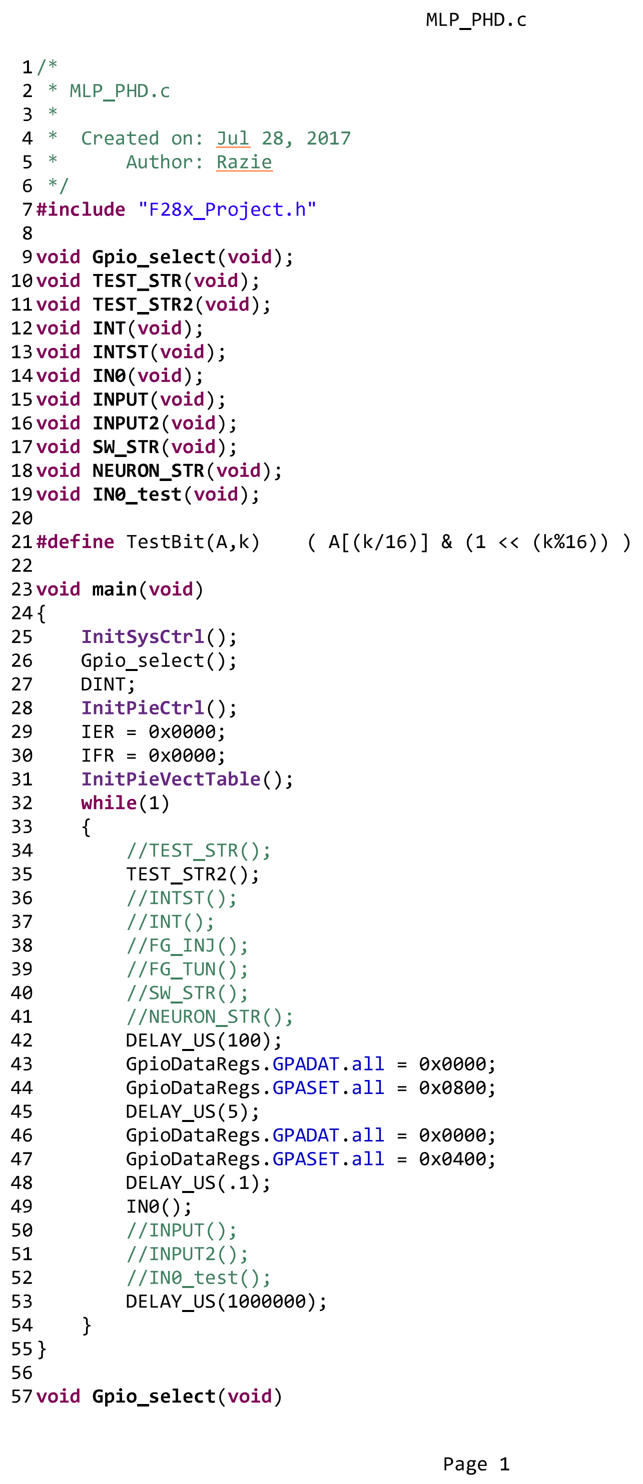
Figure A.5: Simulation Code for the Fourth Switch Bitstream.



```
                                   WTA_STR
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111111111 - FFFF
1111111111110110 - 6FFF
1111011011011011 - DB6F
1101101111011011 - DBDB
0110111111111111 - FFF6
1111111111111111 - FFFF
1111111111111111 - FFFF
```

Page 1

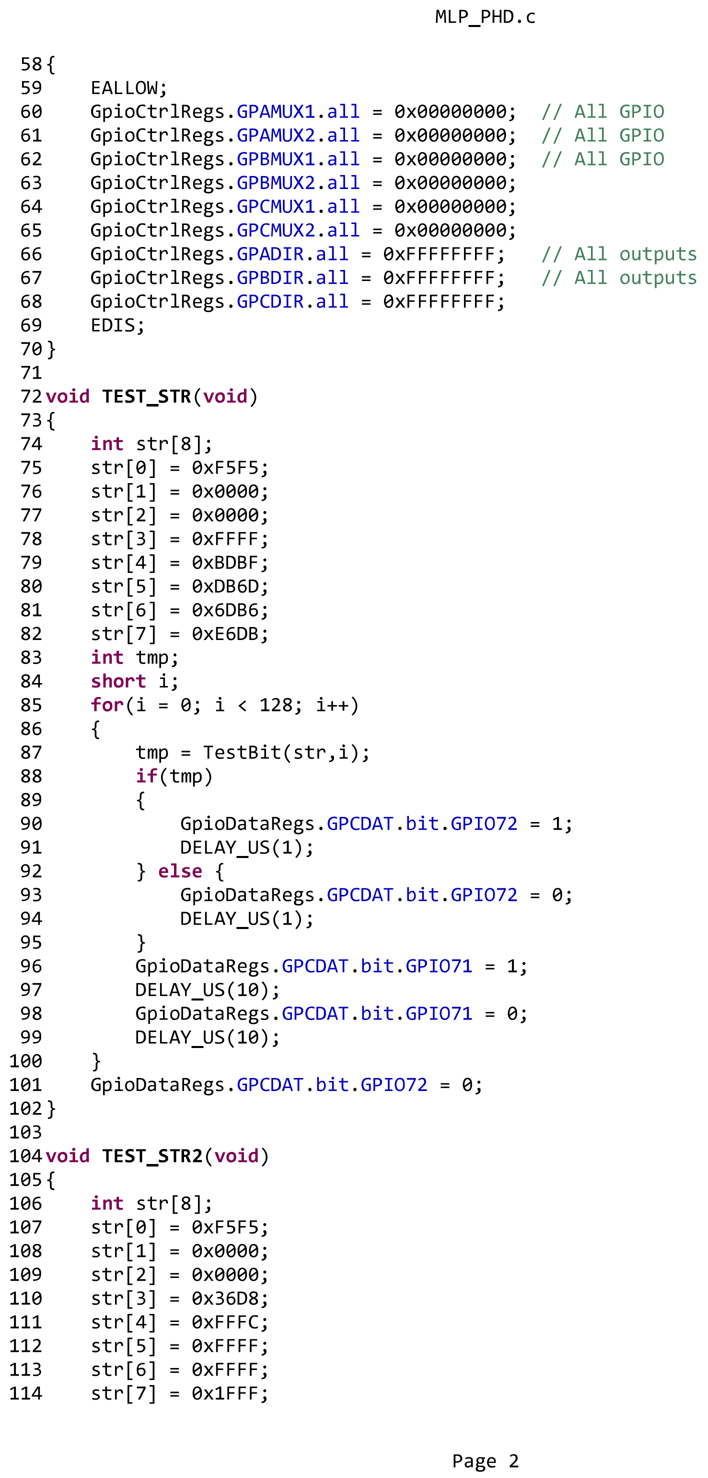
Figure A.6: Simulation Code for Winner-Take-All Bitstream.


MLP_PHD.c

```c
/*
 * MLP_PHD.c
 *
 *  Created on: Jul 28, 2017
 *      Author: Razie
 */
#include "F28x_Project.h"

void Gpio_select(void);
void TEST_STR(void);
void TEST_STR2(void);
void INT(void);
void INTST(void);
void IN0(void);
void INPUT(void);
void INPUT2(void);
void SW_STR(void);
void NEURON_STR(void);
void IN0_test(void);

#define TestBit(A,k)    ( A[(k/16)] & (1 << (k%16)) )

void main(void)
{
    InitSysCtrl();
    Gpio_select();
    DINT;
    InitPieCtrl();
    IER = 0x0000;
    IFR = 0x0000;
    InitPieVectTable();
    while(1)
    {
        //TEST_STR();
        TEST_STR2();
        //INTST();
        //INT();
        //FG_INJ();
        //FG_TUN();
        //SW_STR();
        //NEURON_STR();
        DELAY_US(100);
        GpioDataRegs.GPADAT.all = 0x0000;
        GpioDataRegs.GPASET.all = 0x0800;
        DELAY_US(5);
        GpioDataRegs.GPADAT.all = 0x0000;
        GpioDataRegs.GPASET.all = 0x0400;
        DELAY_US(.1);
        IN0();
        //INPUT();
        //INPUT2();
        //IN0_test();
        DELAY_US(1000000);
    }
}

void Gpio_select(void)
```

Page 1

Figure A.7: Page 1 of Microcontroller Code for Measurements.

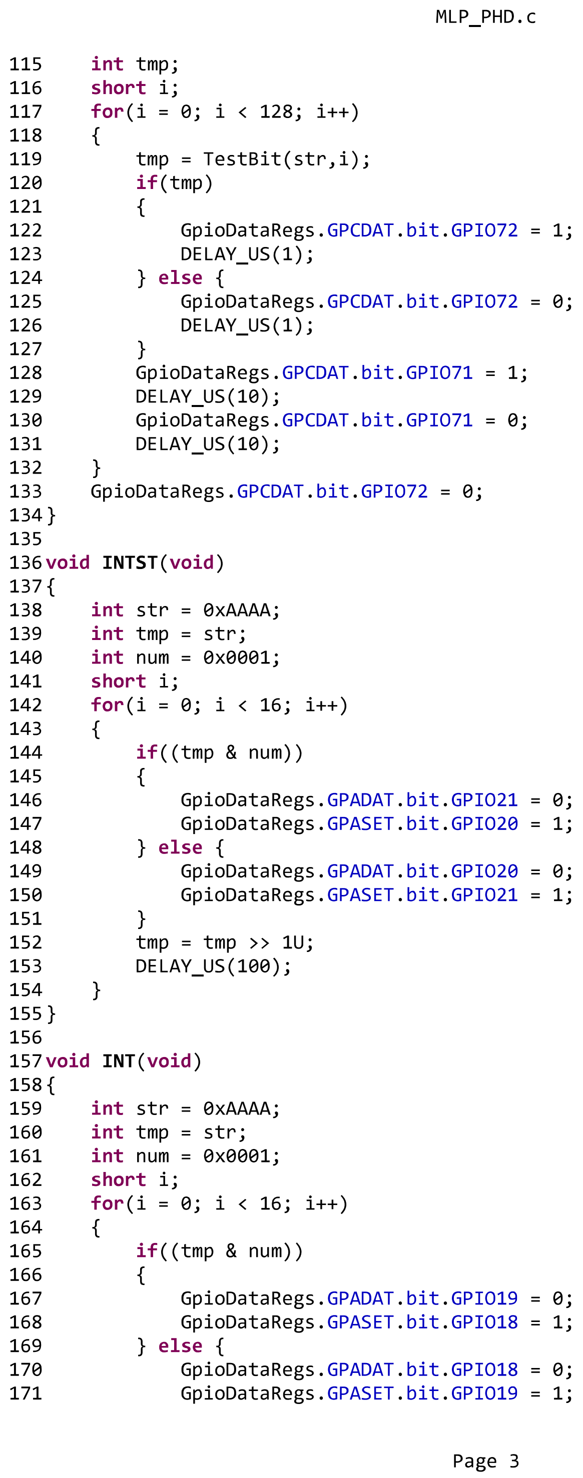


MLP_PHD.c

```c
{
    EALLOW;
    GpioCtrlRegs.GPAMUX1.all = 0x00000000;  // All GPIO
    GpioCtrlRegs.GPAMUX2.all = 0x00000000;  // All GPIO
    GpioCtrlRegs.GPBMUX1.all = 0x00000000;  // All GPIO
    GpioCtrlRegs.GPBMUX2.all = 0x00000000;
    GpioCtrlRegs.GPCMUX1.all = 0x00000000;
    GpioCtrlRegs.GPCMUX2.all = 0x00000000;
    GpioCtrlRegs.GPADIR.all = 0xFFFFFFFF;   // All outputs
    GpioCtrlRegs.GPBDIR.all = 0xFFFFFFFF;   // All outputs
    GpioCtrlRegs.GPCDIR.all = 0xFFFFFFFF;
    EDIS;
}

void TEST_STR(void)
{
    int str[8];
    str[0] = 0xF5F5;
    str[1] = 0x0000;
    str[2] = 0x0000;
    str[3] = 0xFFFF;
    str[4] = 0xBDBF;
    str[5] = 0xDB6D;
    str[6] = 0x6DB6;
    str[7] = 0xE6DB;
    int tmp;
    short i;
    for(i = 0; i < 128; i++)
    {
        tmp = TestBit(str,i);
        if(tmp)
        {
            GpioDataRegs.GPCDAT.bit.GPIO72 = 1;
            DELAY_US(1);
        } else {
            GpioDataRegs.GPCDAT.bit.GPIO72 = 0;
            DELAY_US(1);
        }
        GpioDataRegs.GPCDAT.bit.GPIO71 = 1;
        DELAY_US(10);
        GpioDataRegs.GPCDAT.bit.GPIO71 = 0;
        DELAY_US(10);
    }
    GpioDataRegs.GPCDAT.bit.GPIO72 = 0;
}

void TEST_STR2(void)
{
    int str[8];
    str[0] = 0xF5F5;
    str[1] = 0x0000;
    str[2] = 0x0000;
    str[3] = 0x36D8;
    str[4] = 0xFFFC;
    str[5] = 0xFFFF;
    str[6] = 0xFFFF;
    str[7] = 0x1FFF;
```

Page 2

Figure A.8: Page 2 of Microcontroller Code for Measurements.

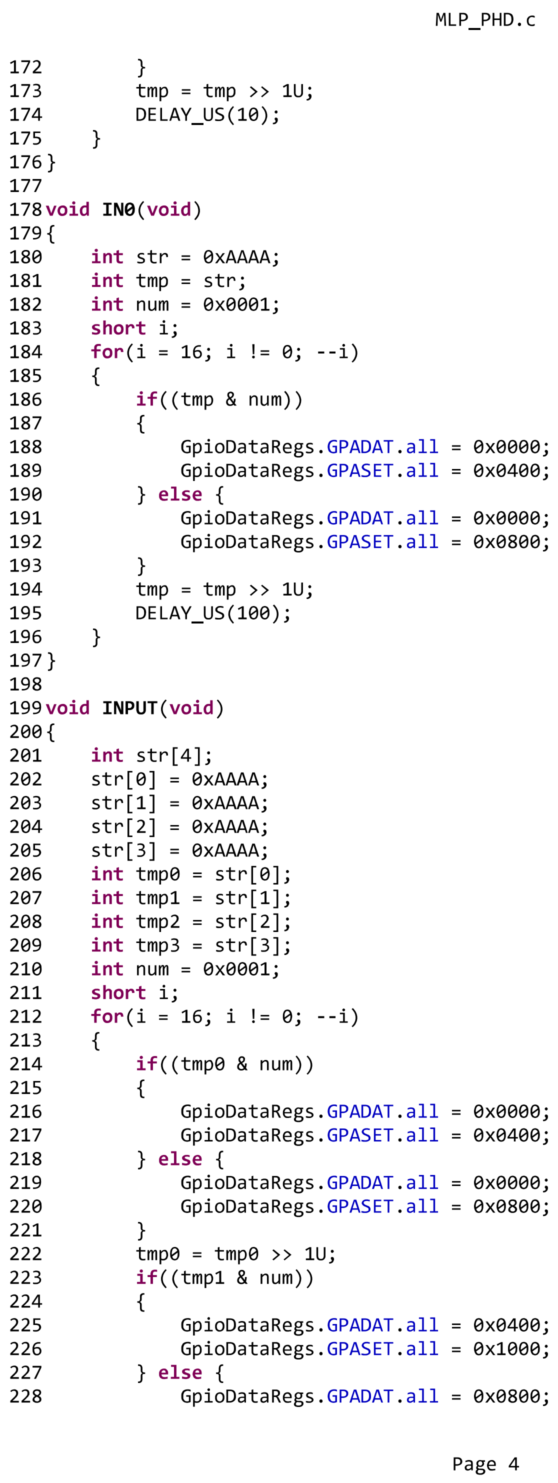


MLP_PHD.c

```c
    int tmp;
    short i;
    for(i = 0; i < 128; i++)
    {
        tmp = TestBit(str,i);
        if(tmp)
        {
            GpioDataRegs.GPCDAT.bit.GPIO72 = 1;
            DELAY_US(1);
        } else {
            GpioDataRegs.GPCDAT.bit.GPIO72 = 0;
            DELAY_US(1);
        }
        GpioDataRegs.GPCDAT.bit.GPIO71 = 1;
        DELAY_US(10);
        GpioDataRegs.GPCDAT.bit.GPIO71 = 0;
        DELAY_US(10);
    }
    GpioDataRegs.GPCDAT.bit.GPIO72 = 0;
}

void INTST(void)
{
    int str = 0xAAAA;
    int tmp = str;
    int num = 0x0001;
    short i;
    for(i = 0; i < 16; i++)
    {
        if((tmp & num))
        {
            GpioDataRegs.GPADAT.bit.GPIO21 = 0;
            GpioDataRegs.GPASET.bit.GPIO20 = 1;
        } else {
            GpioDataRegs.GPADAT.bit.GPIO20 = 0;
            GpioDataRegs.GPASET.bit.GPIO21 = 1;
        }
        tmp = tmp >> 1U;
        DELAY_US(100);
    }
}

void INT(void)
{
    int str = 0xAAAA;
    int tmp = str;
    int num = 0x0001;
    short i;
    for(i = 0; i < 16; i++)
    {
        if((tmp & num))
        {
            GpioDataRegs.GPADAT.bit.GPIO19 = 0;
            GpioDataRegs.GPASET.bit.GPIO18 = 1;
        } else {
            GpioDataRegs.GPADAT.bit.GPIO18 = 0;
            GpioDataRegs.GPASET.bit.GPIO19 = 1;
```

Page 3

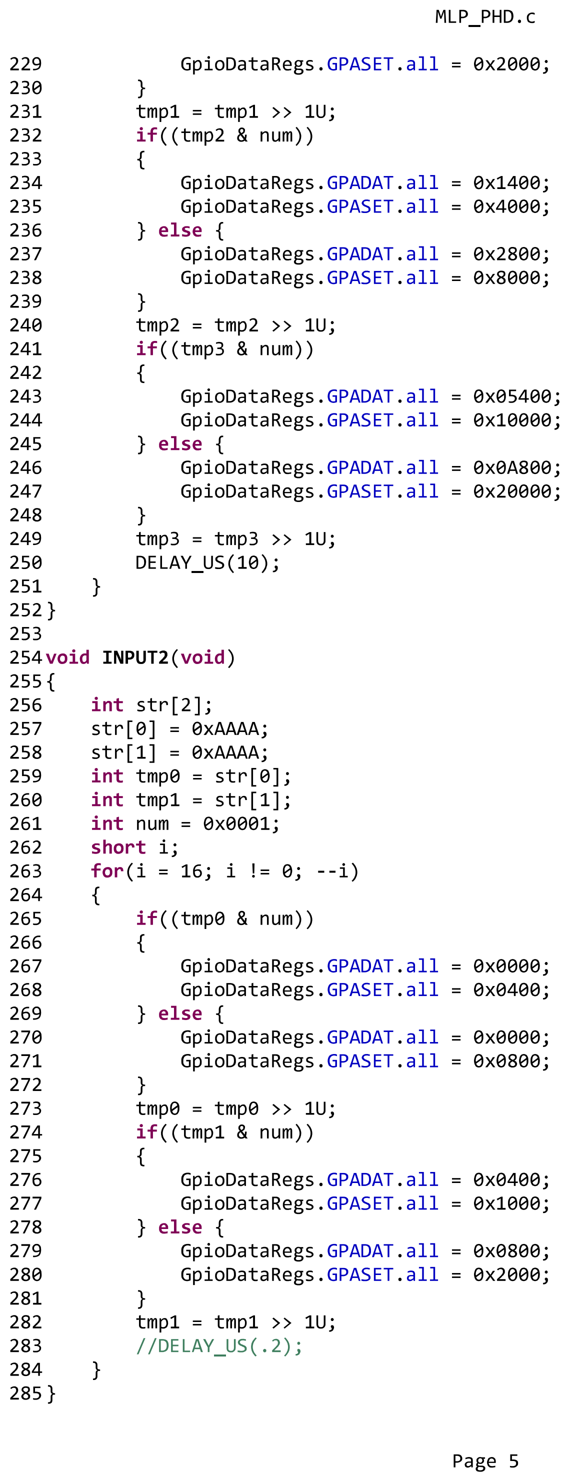
Figure A.9: Page 3 of Microcontroller Code for Measurements.


MLP_PHD.c

```c
        }
        tmp = tmp >> 1U;
        DELAY_US(10);
    }
}

void IN0(void)
{
    int str = 0xAAAA;
    int tmp = str;
    int num = 0x0001;
    short i;
    for(i = 16; i != 0; --i)
    {
        if((tmp & num))
        {
            GpioDataRegs.GPADAT.all = 0x0000;
            GpioDataRegs.GPASET.all = 0x0400;
        } else {
            GpioDataRegs.GPADAT.all = 0x0000;
            GpioDataRegs.GPASET.all = 0x0800;
        }
        tmp = tmp >> 1U;
        DELAY_US(100);
    }
}

void INPUT(void)
{
    int str[4];
    str[0] = 0xAAAA;
    str[1] = 0xAAAA;
    str[2] = 0xAAAA;
    str[3] = 0xAAAA;
    int tmp0 = str[0];
    int tmp1 = str[1];
    int tmp2 = str[2];
    int tmp3 = str[3];
    int num = 0x0001;
    short i;
    for(i = 16; i != 0; --i)
    {
        if((tmp0 & num))
        {
            GpioDataRegs.GPADAT.all = 0x0000;
            GpioDataRegs.GPASET.all = 0x0400;
        } else {
            GpioDataRegs.GPADAT.all = 0x0000;
            GpioDataRegs.GPASET.all = 0x0800;
        }
        tmp0 = tmp0 >> 1U;
        if((tmp1 & num))
        {
            GpioDataRegs.GPADAT.all = 0x0400;
            GpioDataRegs.GPASET.all = 0x1000;
        } else {
            GpioDataRegs.GPADAT.all = 0x0800;
```

Page 4

Figure A.10: Page 4 of Microcontroller Code for Measurements.



MLP_PHD.c

```c
            GpioDataRegs.GPASET.all = 0x2000;
        }
        tmp1 = tmp1 >> 1U;
        if((tmp2 & num))
        {
            GpioDataRegs.GPADAT.all = 0x1400;
            GpioDataRegs.GPASET.all = 0x4000;
        } else {
            GpioDataRegs.GPADAT.all = 0x2800;
            GpioDataRegs.GPASET.all = 0x8000;
        }
        tmp2 = tmp2 >> 1U;
        if((tmp3 & num))
        {
            GpioDataRegs.GPADAT.all = 0x05400;
            GpioDataRegs.GPASET.all = 0x10000;
        } else {
            GpioDataRegs.GPADAT.all = 0x0A800;
            GpioDataRegs.GPASET.all = 0x20000;
        }
        tmp3 = tmp3 >> 1U;
        DELAY_US(10);
    }
}

void INPUT2(void)
{
    int str[2];
    str[0] = 0xAAAA;
    str[1] = 0xAAAA;
    int tmp0 = str[0];
    int tmp1 = str[1];
    int num = 0x0001;
    short i;
    for(i = 16; i != 0; --i)
    {
        if((tmp0 & num))
        {
            GpioDataRegs.GPADAT.all = 0x0000;
            GpioDataRegs.GPASET.all = 0x0400;
        } else {
            GpioDataRegs.GPADAT.all = 0x0000;
            GpioDataRegs.GPASET.all = 0x0800;
        }
        tmp0 = tmp0 >> 1U;
        if((tmp1 & num))
        {
            GpioDataRegs.GPADAT.all = 0x0400;
            GpioDataRegs.GPASET.all = 0x1000;
        } else {
            GpioDataRegs.GPADAT.all = 0x0800;
            GpioDataRegs.GPASET.all = 0x2000;
        }
        tmp1 = tmp1 >> 1U;
        //DELAY_US(.2);
    }
}
```

Page 5

Figure A.11: Page 5 of Microcontroller Code for Measurements.

```

void IN0_test(void)
{
    GpioDataRegs.GPATOGGLE.all = 0x0800;
    GpioDataRegs.GPATOGGLE.all = 0X0400;
    GpioDataRegs.GPATOGGLE.all = 0x0800;
    GpioDataRegs.GPATOGGLE.all = 0X0400;
    GpioDataRegs.GPATOGGLE.all = 0x0800;
    GpioDataRegs.GPATOGGLE.all = 0X0400;
    GpioDataRegs.GPATOGGLE.all = 0x0800;
    GpioDataRegs.GPATOGGLE.all = 0X0400;
    GpioDataRegs.GPATOGGLE.all = 0x0800;
    GpioDataRegs.GPATOGGLE.all = 0X0400;
    GpioDataRegs.GPATOGGLE.all = 0x0800;
    GpioDataRegs.GPATOGGLE.all = 0X0400;
    GpioDataRegs.GPATOGGLE.all = 0x0800;
    GpioDataRegs.GPATOGGLE.all = 0X0400;
    GpioDataRegs.GPATOGGLE.all = 0x0800;
    GpioDataRegs.GPATOGGLE.all = 0X0400;
}

void SW_STR(void)
{
    int str0[15];
    str0[0] = 0xFFFF;
    str0[1] = 0xFFFF;
    str0[2] = 0xFFFF;
    str0[3] = 0xFFFF;
    str0[4] = 0xFFFF;
    str0[5] = 0xFFFF;
    str0[6] = 0xFFFF;
    str0[7] = 0xFFFF;
    str0[8] = 0xFF1F;
    str0[9] = 0xFFFF;
    str0[10] = 0xFF7F;
    str0[11] = 0xFFFF;
    str0[12] = 0xFFFF;
    str0[13] = 0xFFFF;
    str0[14] = 0xFFFF;
    int str1[15];
    str1[0] = 0xFFFF;
    str1[1] = 0xFFFF;
    str1[2] = 0xFFFF;
    str1[3] = 0xFFFF;
    str1[4] = 0xFFFF;
    str1[5] = 0xFFFF;
    str1[6] = 0xFFFF;
    str1[7] = 0xFFFF;
    str1[8] = 0xFFFF;
    str1[9] = 0xFFFF;
    str1[10] = 0xFEFE;
    str1[11] = 0xFFFF;
    str1[12] = 0xFFFF;
    str1[13] = 0xFFFF;
    str1[14] = 0xFFFF;
    int str2[15];
    str2[0] = 0xFFFF;
```

Figure A.12: Page 6 of Microcontroller Code for Measurements.

```
str2[1] = 0xFFFF;
str2[2] = 0xFFFF;
str2[3] = 0xFFFF;
str2[4] = 0xFFFF;
str2[5] = 0xFFFF;
str2[6] = 0xFFFF;
str2[7] = 0xFFFF;
str2[8] = 0xFFFF;
str2[9] = 0xFFFF;
str2[10] = 0xFEFE;
str2[11] = 0xFFFF;
str2[12] = 0xFFFF;
str2[13] = 0xFFFF;
str2[14] = 0xFFFF;
int str3[15];
str3[0] = 0xFFFF;
str3[1] = 0xFFFF;
str3[2] = 0xFFFF;
str3[3] = 0xFFFF;
str3[4] = 0xFFFF;
str3[5] = 0xFFFF;
str3[6] = 0xFFFF;
str3[7] = 0xFFFF;
str3[8] = 0xFFFF;
str3[9] = 0xFFFF;
str3[10] = 0xFFFF;
str3[11] = 0xF3FF;
str3[12] = 0xF7FF;
str3[13] = 0xFFFF;
str3[14] = 0xFFFF;
int tmp0;
int tmp1;
int tmp2;
int tmp3;
short i;
for(i = 0; i < 240; i++)
{
    tmp0 = TestBit(str0,i);
    tmp1 = TestBit(str1,i);
    tmp2 = TestBit(str2,i);
    tmp3 = TestBit(str3,i);
    if(tmp0)
    {
        GpioDataRegs.GPBDAT.bit.GPIO58 = 1;
        DELAY_US(1);
    } else {
        GpioDataRegs.GPBDAT.bit.GPIO58 = 0;
        DELAY_US(1);
    }
    if(tmp1)
    {
        GpioDataRegs.GPBDAT.bit.GPIO59 = 1;
        DELAY_US(1);
    } else {
        GpioDataRegs.GPBDAT.bit.GPIO59 = 0;
        DELAY_US(1);
    }
```

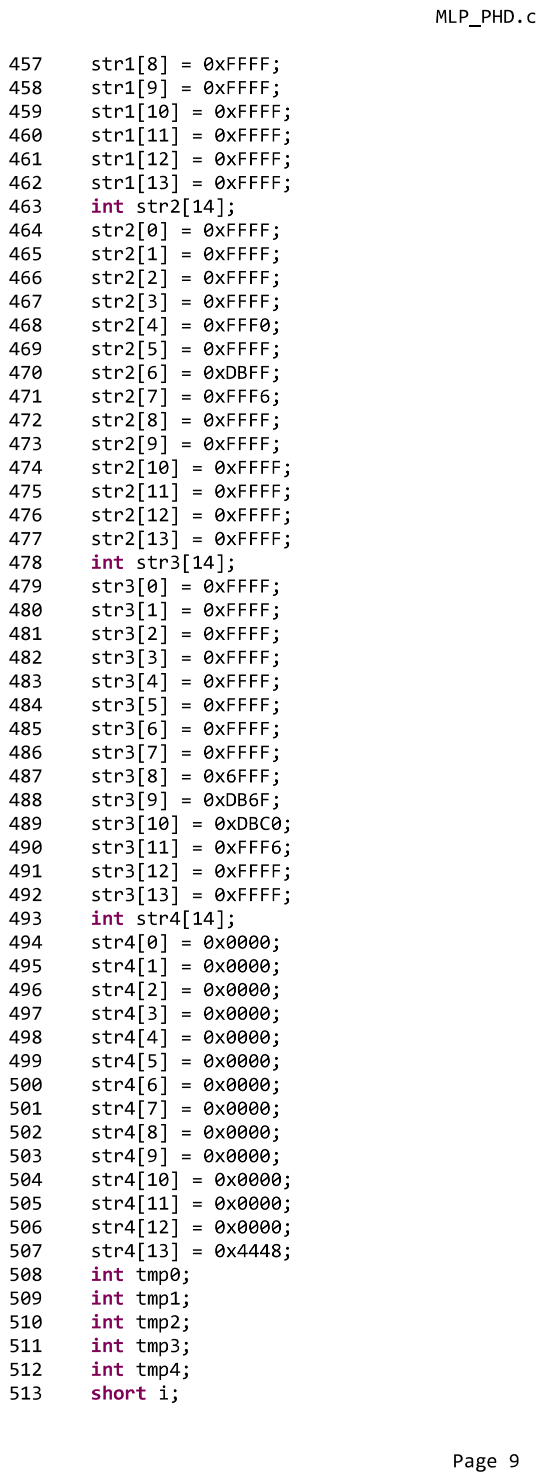
Figure A.13: Page 7 of Microcontroller Code for Measurements.

MLP_PHD.c

```c
        if(tmp2)
        {
            GpioDataRegs.GPBDAT.bit.GPIO60 = 1;
            DELAY_US(1);
        } else {
            GpioDataRegs.GPBDAT.bit.GPIO60 = 0;
            DELAY_US(1);
        }
        if(tmp3)
        {
            GpioDataRegs.GPBDAT.bit.GPIO61 = 1;
            DELAY_US(1);
        } else {
            GpioDataRegs.GPBDAT.bit.GPIO61 = 0;
            DELAY_US(1);
        }
        GpioDataRegs.GPBDAT.bit.GPIO42 = 1;
        DELAY_US(10);
        GpioDataRegs.GPBDAT.bit.GPIO42 = 0;
        DELAY_US(10);
    }
    GpioDataRegs.GPBDAT.bit.GPIO61 = 0;
    DELAY_US(10);
    GpioDataRegs.GPBDAT.bit.GPIO60 = 0;
    DELAY_US(10);
    GpioDataRegs.GPBDAT.bit.GPIO59 = 0;
    DELAY_US(10);
    GpioDataRegs.GPBDAT.bit.GPIO58 = 0;
    DELAY_US(10);
}

void NEURON_STR(void)
{
    int str0[14];
    str0[0] = 0xFFFF;
    str0[1] = 0xFFFF;
    str0[2] = 0xFFFF;
    str0[3] = 0xFFFF;
    str0[4] = 0xFFF0;
    str0[5] = 0xFFFF;
    str0[6] = 0xDBFF;
    str0[7] = 0xFFF6;
    str0[8] = 0xFFFF;
    str0[9] = 0xFFFF;
    str0[10] = 0xFFFF;
    str0[11] = 0xFFFF;
    str0[12] = 0xFFFF;
    str0[13] = 0xFFFF;
    int str1[14];
    str1[0] = 0xFFFF;
    str1[1] = 0xFFFF;
    str1[2] = 0xFFFF;
    str1[3] = 0xFFFF;
    str1[4] = 0xFFF0;
    str1[5] = 0xFFFF;
    str1[6] = 0xDBFF;
    str1[7] = 0xFFF6;
```

Page 8

Figure A.14: Page 8 of Microcontroller Code for Measurements.



MLP_PHD.c

```c
str1[8] = 0xFFFF;
str1[9] = 0xFFFF;
str1[10] = 0xFFFF;
str1[11] = 0xFFFF;
str1[12] = 0xFFFF;
str1[13] = 0xFFFF;
int str2[14];
str2[0] = 0xFFFF;
str2[1] = 0xFFFF;
str2[2] = 0xFFFF;
str2[3] = 0xFFFF;
str2[4] = 0xFFF0;
str2[5] = 0xFFFF;
str2[6] = 0xDBFF;
str2[7] = 0xFFF6;
str2[8] = 0xFFFF;
str2[9] = 0xFFFF;
str2[10] = 0xFFFF;
str2[11] = 0xFFFF;
str2[12] = 0xFFFF;
str2[13] = 0xFFFF;
int str3[14];
str3[0] = 0xFFFF;
str3[1] = 0xFFFF;
str3[2] = 0xFFFF;
str3[3] = 0xFFFF;
str3[4] = 0xFFFF;
str3[5] = 0xFFFF;
str3[6] = 0xFFFF;
str3[7] = 0xFFFF;
str3[8] = 0x6FFF;
str3[9] = 0xDB6F;
str3[10] = 0xDBC0;
str3[11] = 0xFFF6;
str3[12] = 0xFFFF;
str3[13] = 0xFFFF;
int str4[14];
str4[0] = 0x0000;
str4[1] = 0x0000;
str4[2] = 0x0000;
str4[3] = 0x0000;
str4[4] = 0x0000;
str4[5] = 0x0000;
str4[6] = 0x0000;
str4[7] = 0x0000;
str4[8] = 0x0000;
str4[9] = 0x0000;
str4[10] = 0x0000;
str4[11] = 0x0000;
str4[12] = 0x0000;
str4[13] = 0x4448;
int tmp0;
int tmp1;
int tmp2;
int tmp3;
int tmp4;
short i;
```

Page 9

Figure A.15: Page 9 of Microcontroller Code for Measurements.

```
    for(i = 0; i < 224; i++)
    {
        tmp0 = TestBit(str0,i);
        tmp1 = TestBit(str1,i);
        tmp2 = TestBit(str2,i);
        tmp3 = TestBit(str3,i);
        tmp4 = TestBit(str4,i);
        if(tmp0)
        {
            GpioDataRegs.GPCDAT.bit.GPIO64 = 1;
            DELAY_US(1);
        } else {
            GpioDataRegs.GPCDAT.bit.GPIO64 = 0;
            DELAY_US(1);
        }
        if(tmp1)
        {
            GpioDataRegs.GPCDAT.bit.GPIO65 = 1;
            DELAY_US(1);
        } else {
            GpioDataRegs.GPCDAT.bit.GPIO65 = 0;
            DELAY_US(1);
        }
        if(tmp2)
        {
            GpioDataRegs.GPCDAT.bit.GPIO66 = 1;
            DELAY_US(1);
        } else {
            GpioDataRegs.GPCDAT.bit.GPIO66 = 0;
            DELAY_US(1);
        }
        if(tmp3)
        {
            GpioDataRegs.GPBDAT.bit.GPIO62 = 1;
            DELAY_US(1);
        } else {
            GpioDataRegs.GPBDAT.bit.GPIO62 = 0;
            DELAY_US(1);
        }
        if(tmp4)
        {
            GpioDataRegs.GPBDAT.bit.GPIO63 = 1;
            DELAY_US(1);
        } else {
            GpioDataRegs.GPBDAT.bit.GPIO63 = 0;
            DELAY_US(1);
        }
        GpioDataRegs.GPBDAT.bit.GPIO43 = 1;
        DELAY_US(10);
        GpioDataRegs.GPBDAT.bit.GPIO43 = 0;
        DELAY_US(10);
    }
    GpioDataRegs.GPCDAT.bit.GPIO64 = 0;
    DELAY_US(10);
    GpioDataRegs.GPCDAT.bit.GPIO65 = 0;
    DELAY_US(10);
    GpioDataRegs.GPCDAT.bit.GPIO66 = 0;
```

Figure A.16: Page 10 of Microcontroller Code for Measurements.

MLP_PHD.c

```c
    DELAY_US(10);
    GpioDataRegs.GPBDAT.bit.GPIO62 = 0;
    DELAY_US(10);
    GpioDataRegs.GPBDAT.bit.GPIO63 = 0;
}

```

Page 11

Figure A.17: Page 11 of Microcontroller Code for Measurements.

Appendix B

MLP System Measurement Results Figures

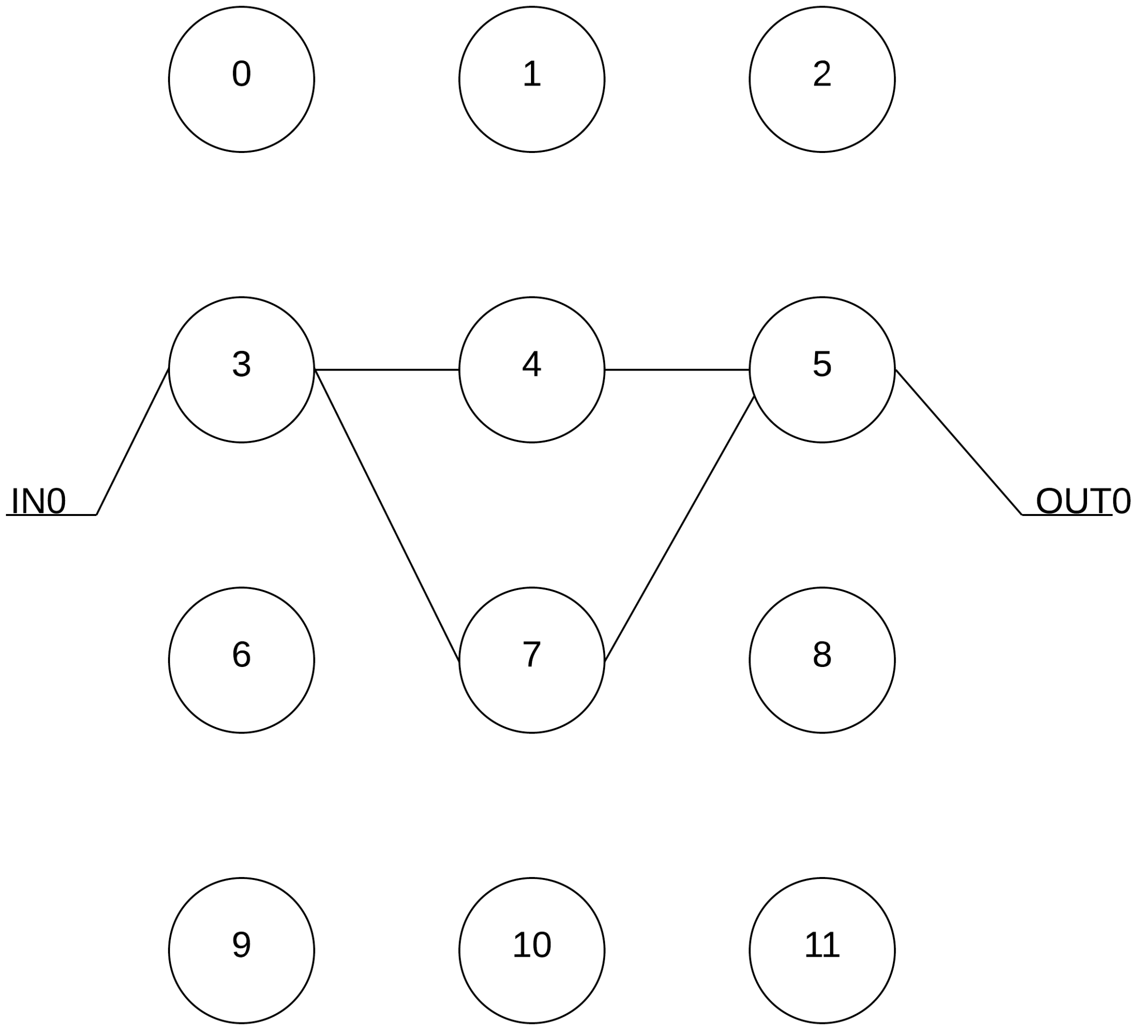
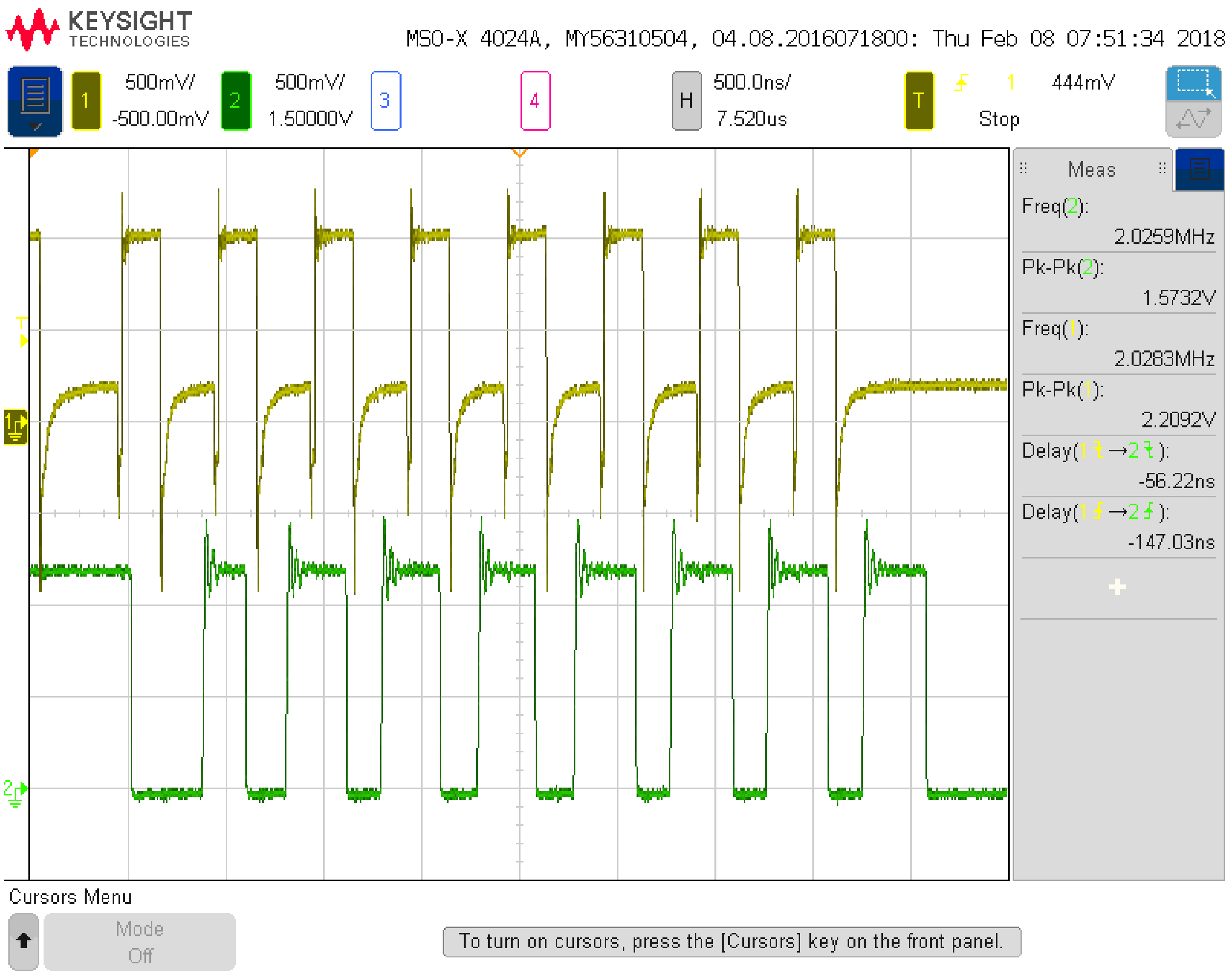


Figure B.1: Second MLP Configuration for Measurement Results.


KEYSIGHT TECHNOLOGIES
MSO-X 4024A, MY56310504, 04.08.2016071800: Thu Feb 08 07:51:34 2018
1 500mV/ -500.00mV
2 500mV/ 1.50000V
3
4
H 500.0ns/ 7.520us
T 1 444mV Stop
Meas
Cursors Menu
Mode Off
To turn on cursors, press the [Cursors] key on the front panel.

Freq(2): 2.0259MHz
Pk-Pk(2): 1.5732V
Freq(1): 2.0283MHz
Pk-Pk(1): 2.2092V
Delay(1→2): -56.22ns
Delay(1→2): -147.03ns

Figure B.2: Second MLP System Configuration Measurement.

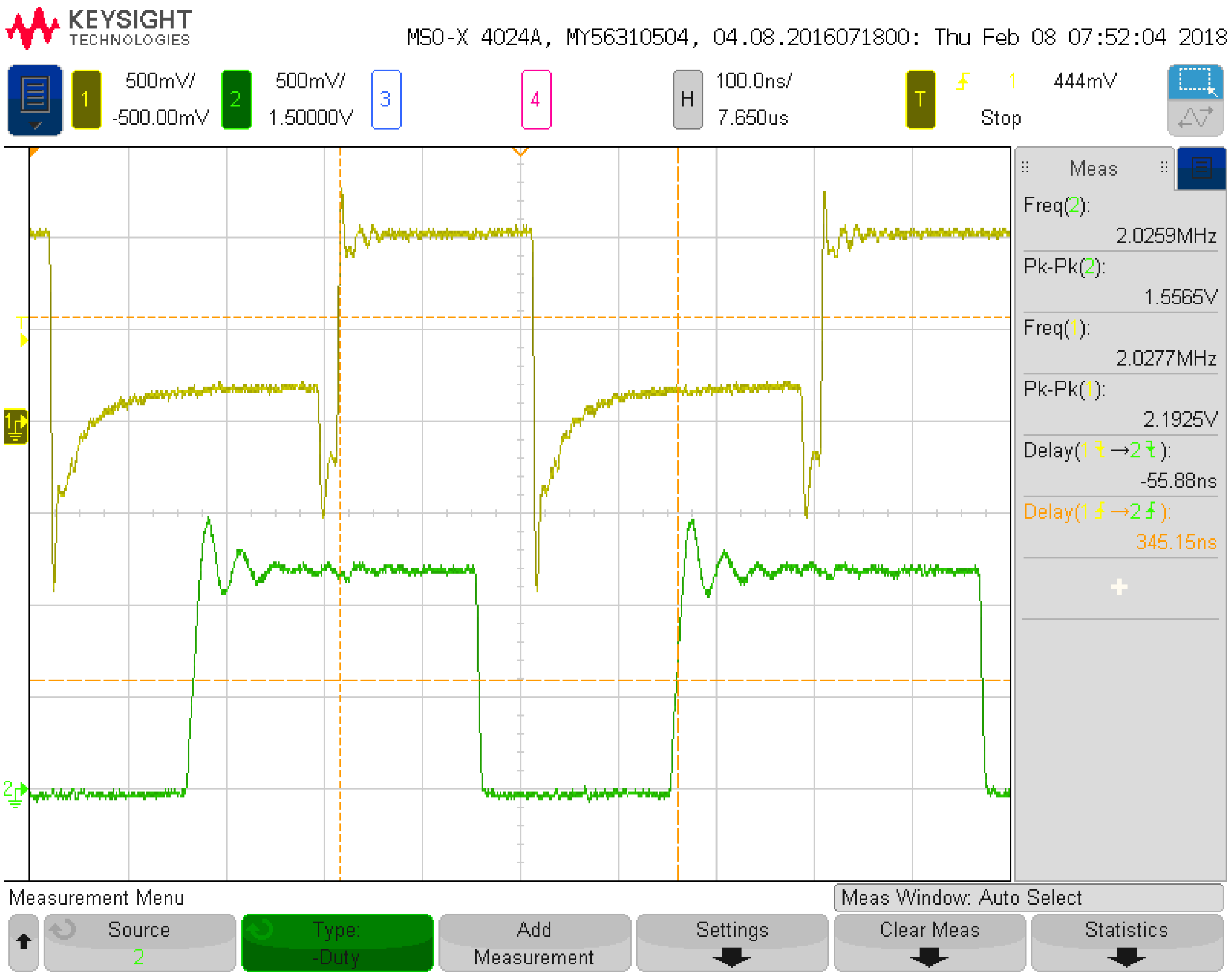


Figure B.3: Second MLP System Configuration Measurement for Rising Edge Propagation Delay.

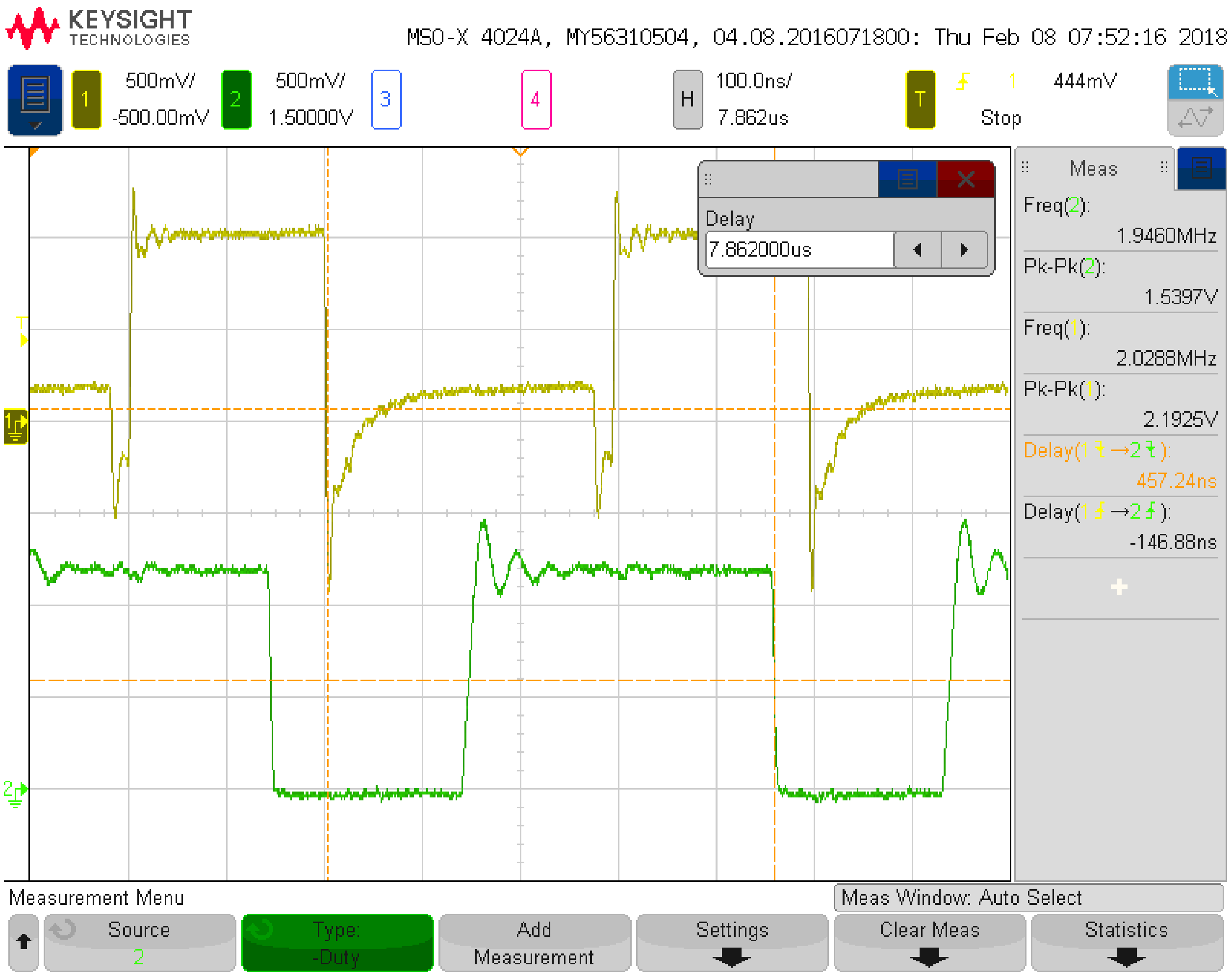


Figure B.4: Second MLP System Configuration Measurement for Falling Edge Propagation Delay.

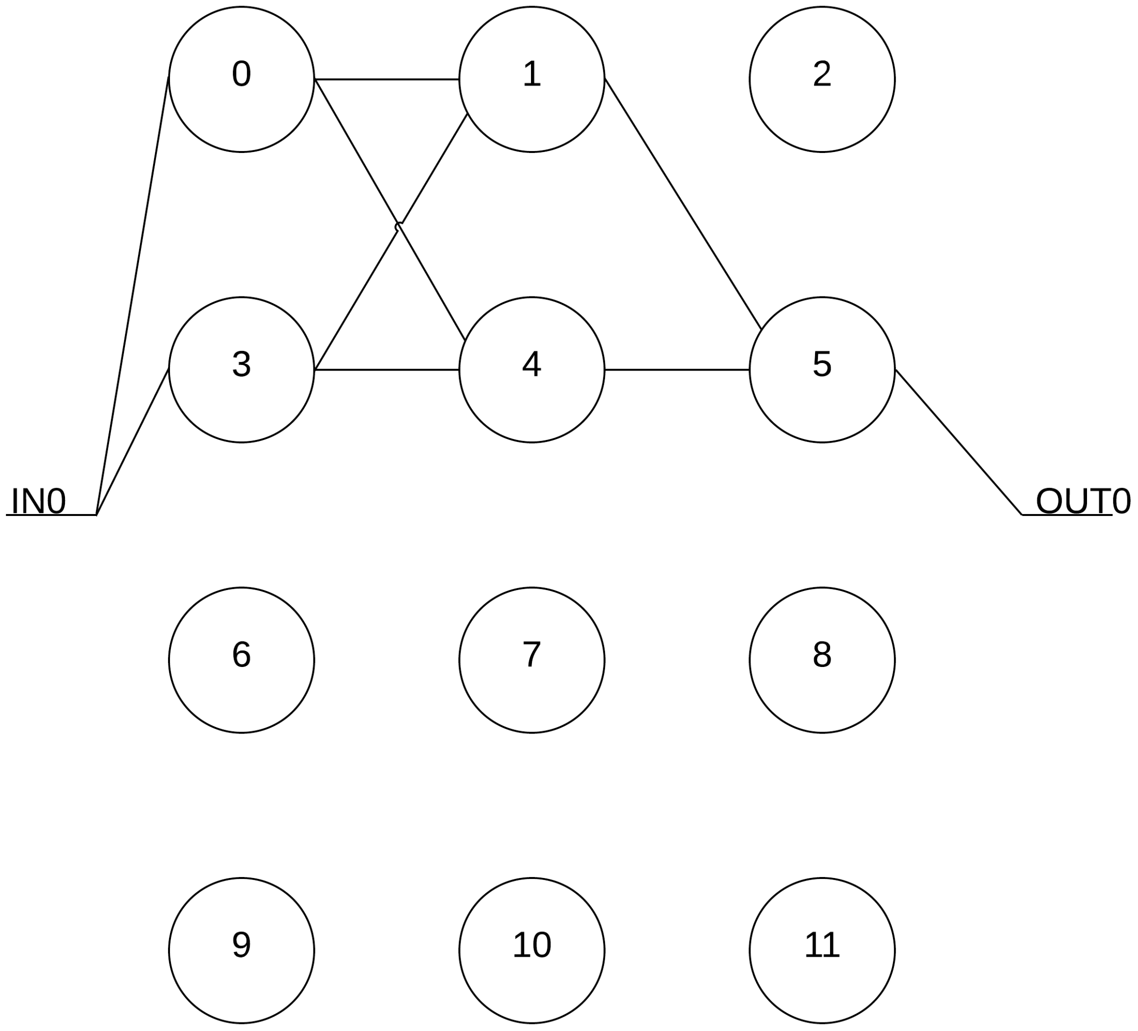
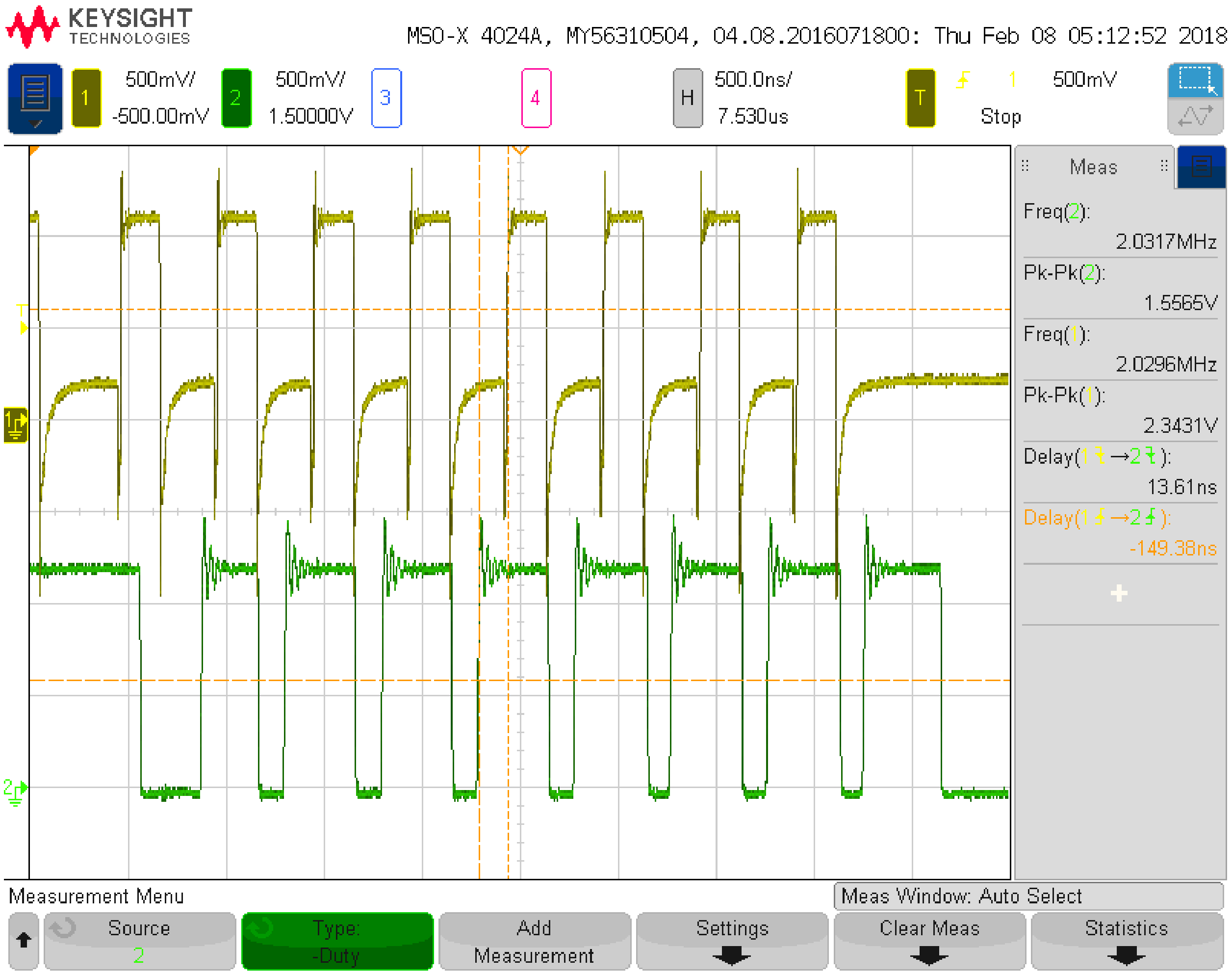
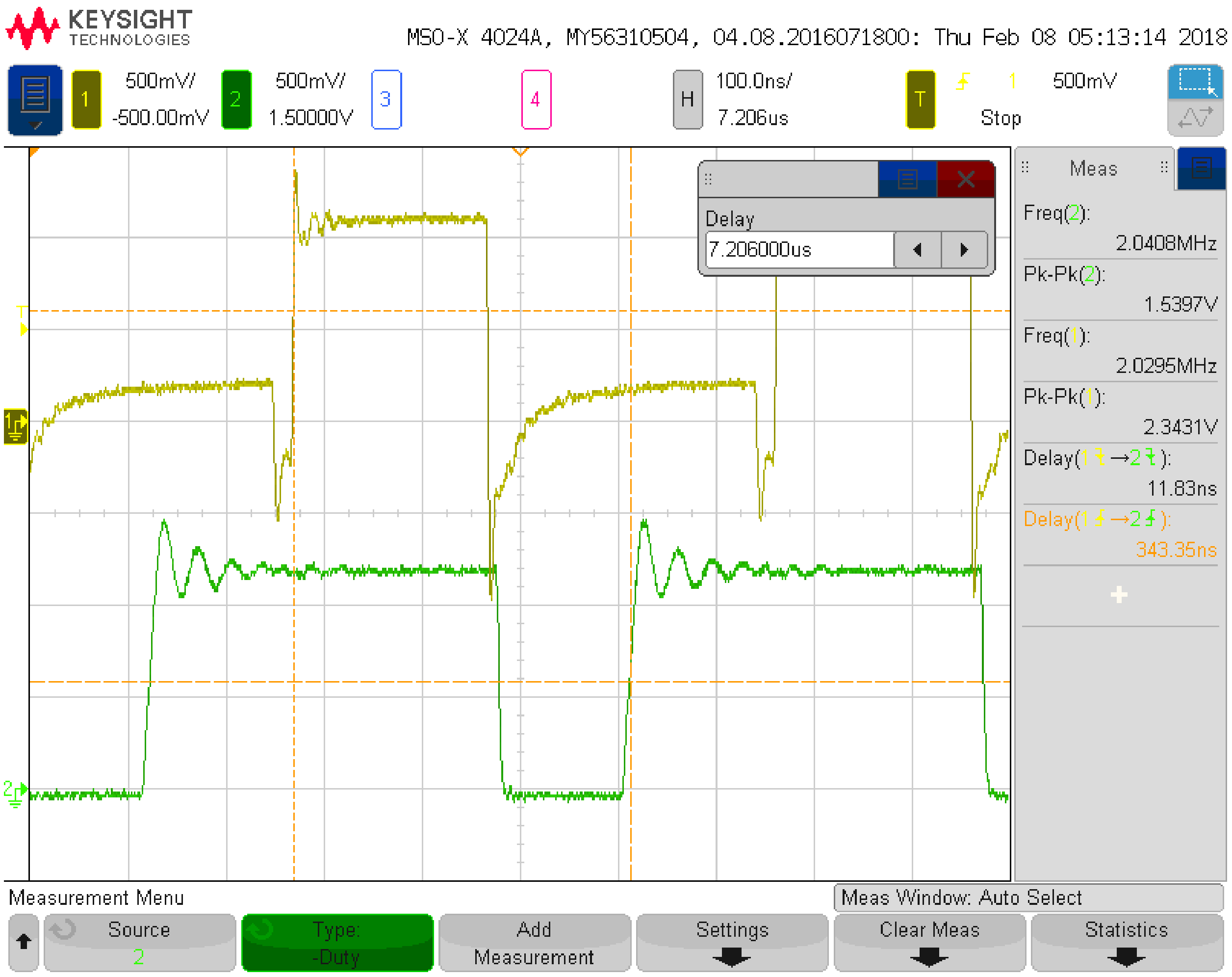


Figure B.5: Third MLP Configuration for Measurement Results.



Freq(2): 2.0317MHz
Pk-Pk(2): 1.5565V
Freq(1): 2.0296MHz
Pk-Pk(1): 2.3431V
Delay(1→2): 13.61ns
Delay(1→2): -149.38ns

Figure B.6: Third MLP System Configuration Measurement.



Freq(2): 2.0408MHz
Pk-Pk(2): 1.5397V
Freq(1): 2.0295MHz
Pk-Pk(1): 2.3431V
Delay(1 → 2): 11.83ns
Delay(1 → 2): 343.35ns

Figure B.7: Third MLP System Configuration Measurement for Rising Edge Propagation Delay.

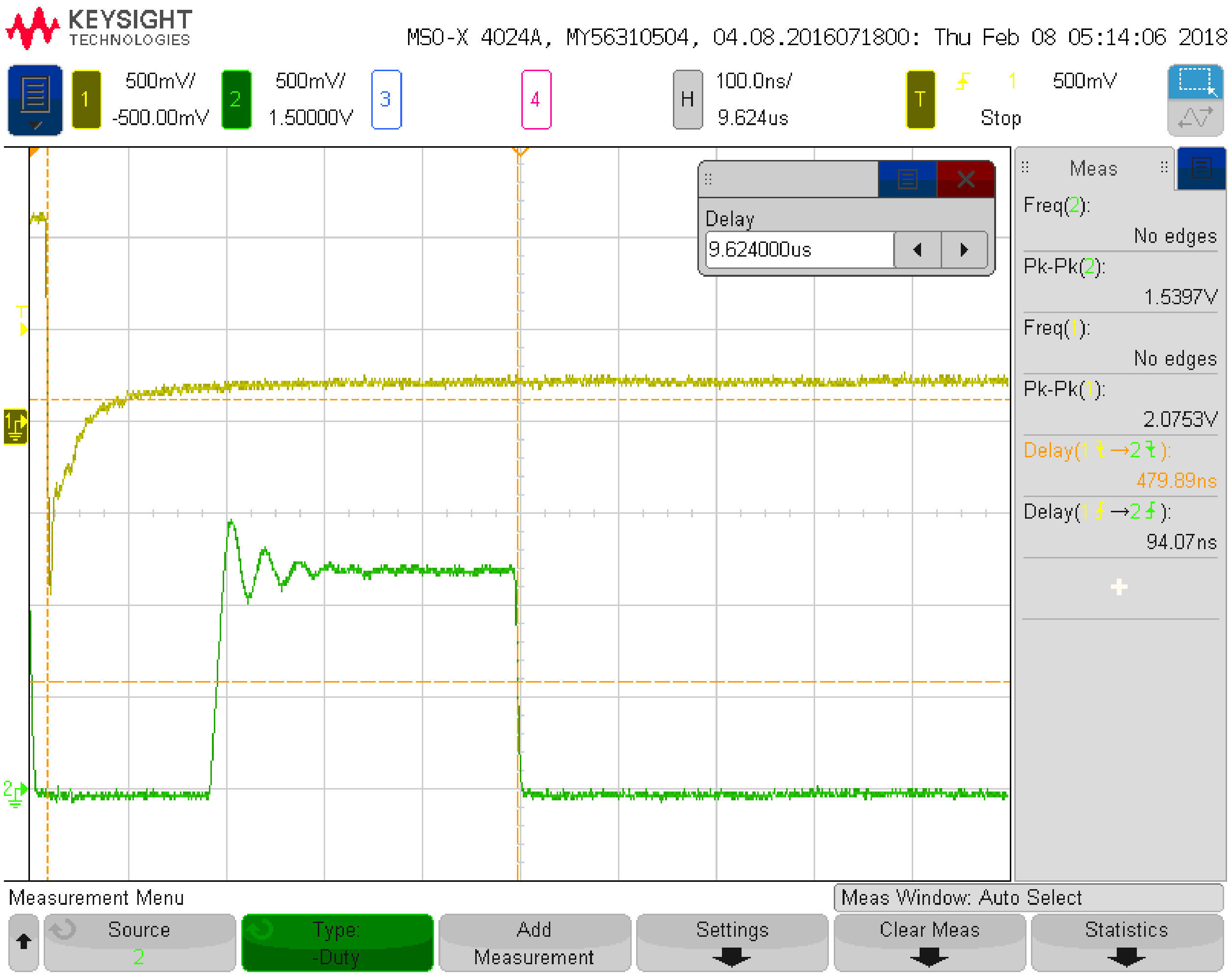


Figure B.8: Third MLP System Configuration Measurement for Falling Edge Propagation Delay.

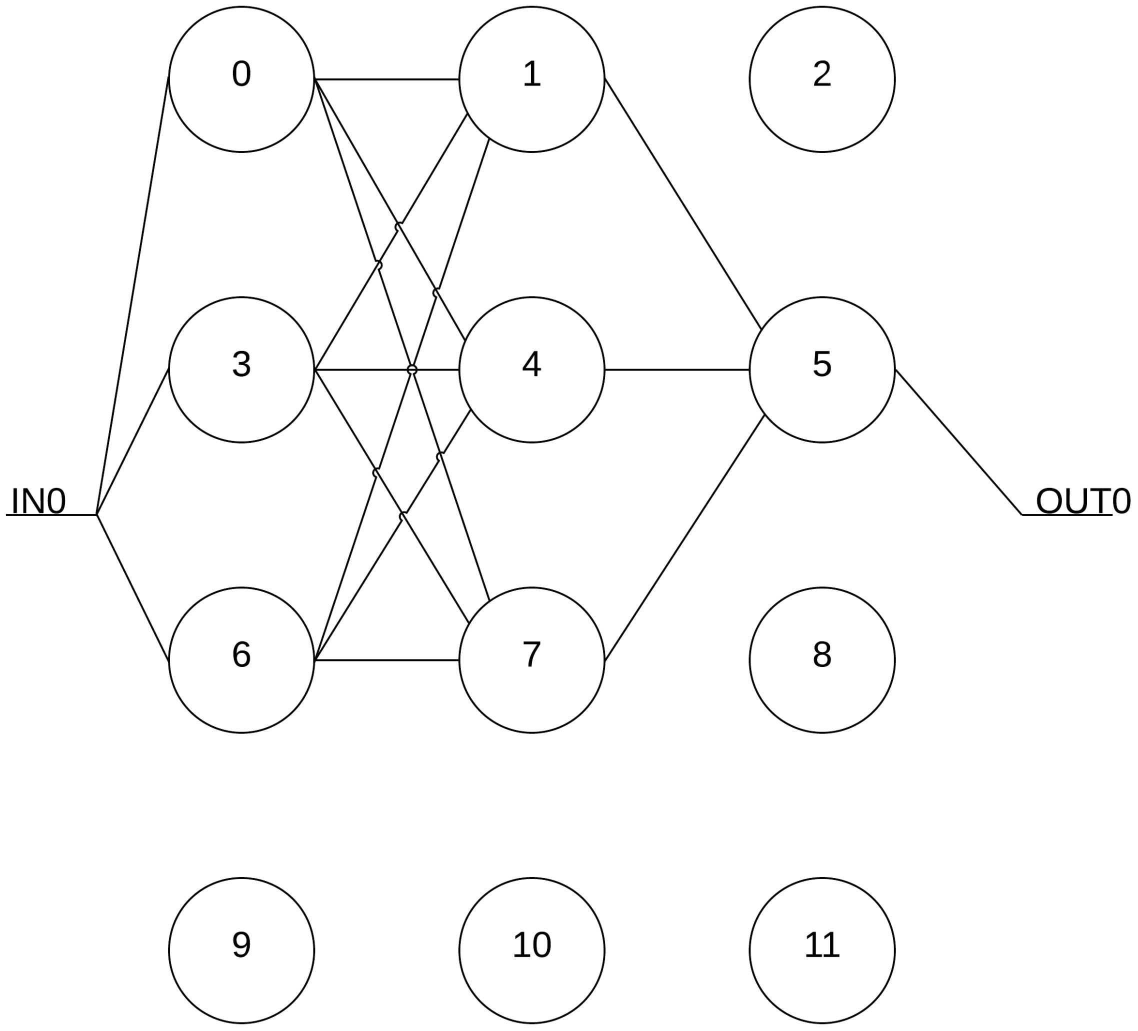
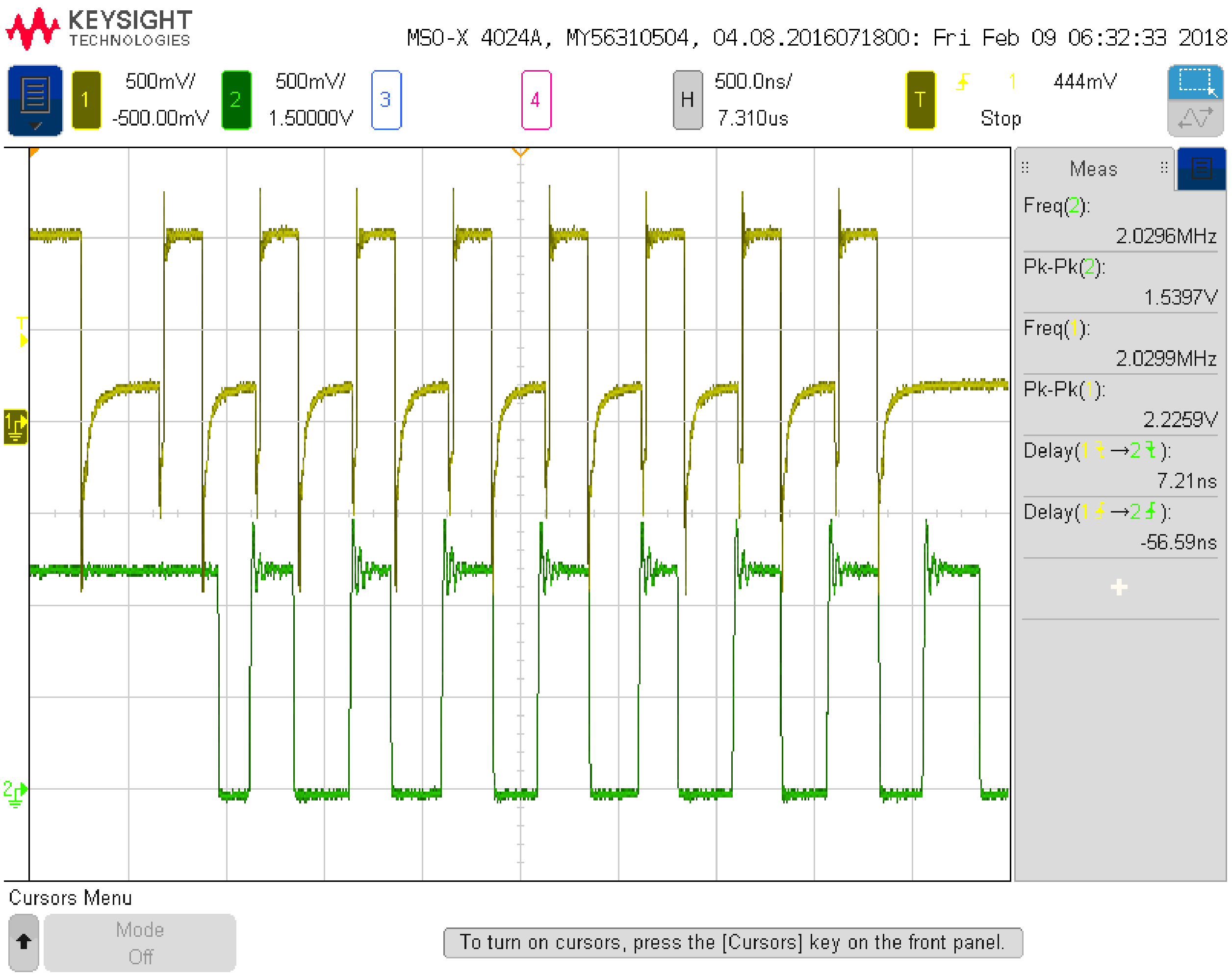


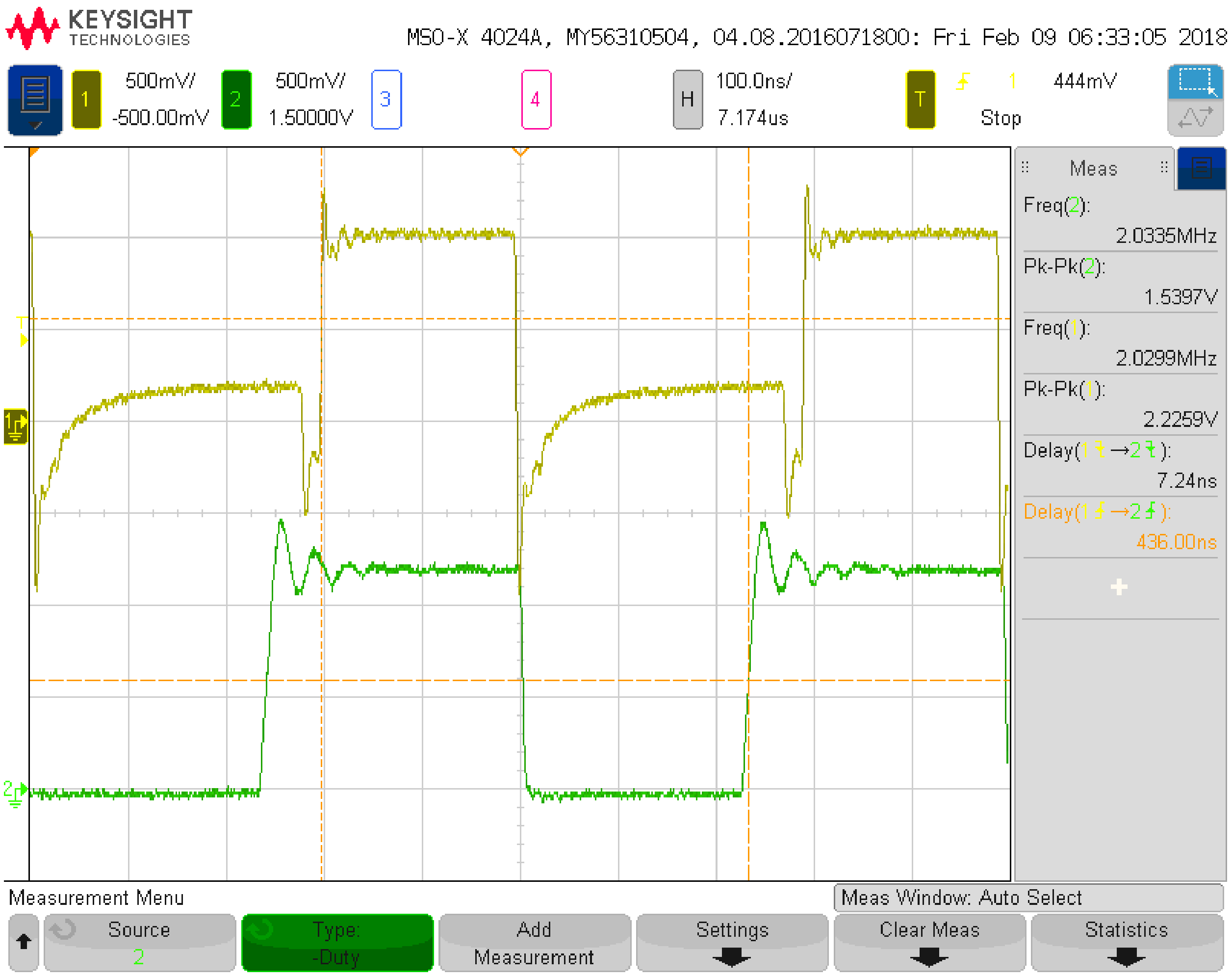
Figure B.9: Fourth MLP Configuration for Measurement Results.



Freq(2): 2.0296MHz
Pk-Pk(2): 1.5397V
Freq(1): 2.0299MHz
Pk-Pk(1): 2.2259V
Delay(1↑→2↑): 7.21ns
Delay(1↓→2↓): -56.59ns

To turn on cursors, press the [Cursors] key on the front panel.

Figure B.10: Fourth MLP System Configuration Measurement.



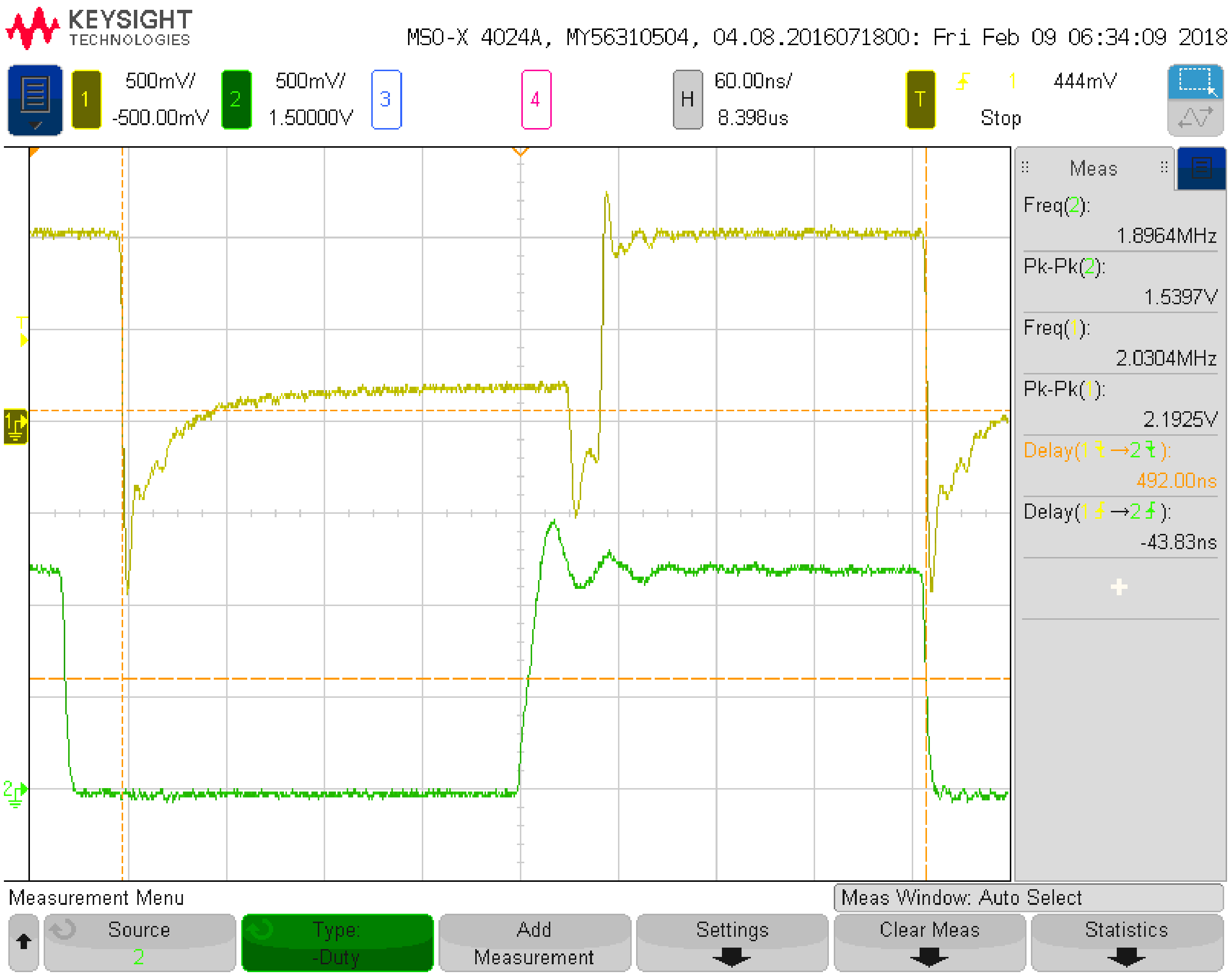
MSO-X 4024A, MY56310504, 04.08.2016071800: Fri Feb 09 06:33:05 2018

500mV/ -500.00mV
500mV/ 1.50000V
100.0ns/ 7.174us
1 444mV Stop

Freq(2): 2.0335MHz
Pk-Pk(2): 1.5397V
Freq(1): 2.0299MHz
Pk-Pk(1): 2.2259V
Delay(1↑→2↑): 7.24ns
Delay(1↓→2↓): 436.00ns

Meas Window: Auto Select

Figure B.11: Fourth MLP System Configuration Measurement for Rising Edge Propagation Delay.



MSO-X 4024A, MY56310504, 04.08.2016071800: Fri Feb 09 06:34:09 2018

1 500mV/ -500.00mV
2 500mV/ 1.50000V
H 60.00ns/ 8.398us
T 1 444mV Stop

Freq(2): 1.8964MHz
Pk-Pk(2): 1.5397V
Freq(1): 2.0304MHz
Pk-Pk(1): 2.1925V
Delay(1 → 2): 492.00ns
Delay(1 → 2): -43.83ns

Figure B.12: Fourth MLP System Configuration Measurement for Falling Edge Propagation Delay.

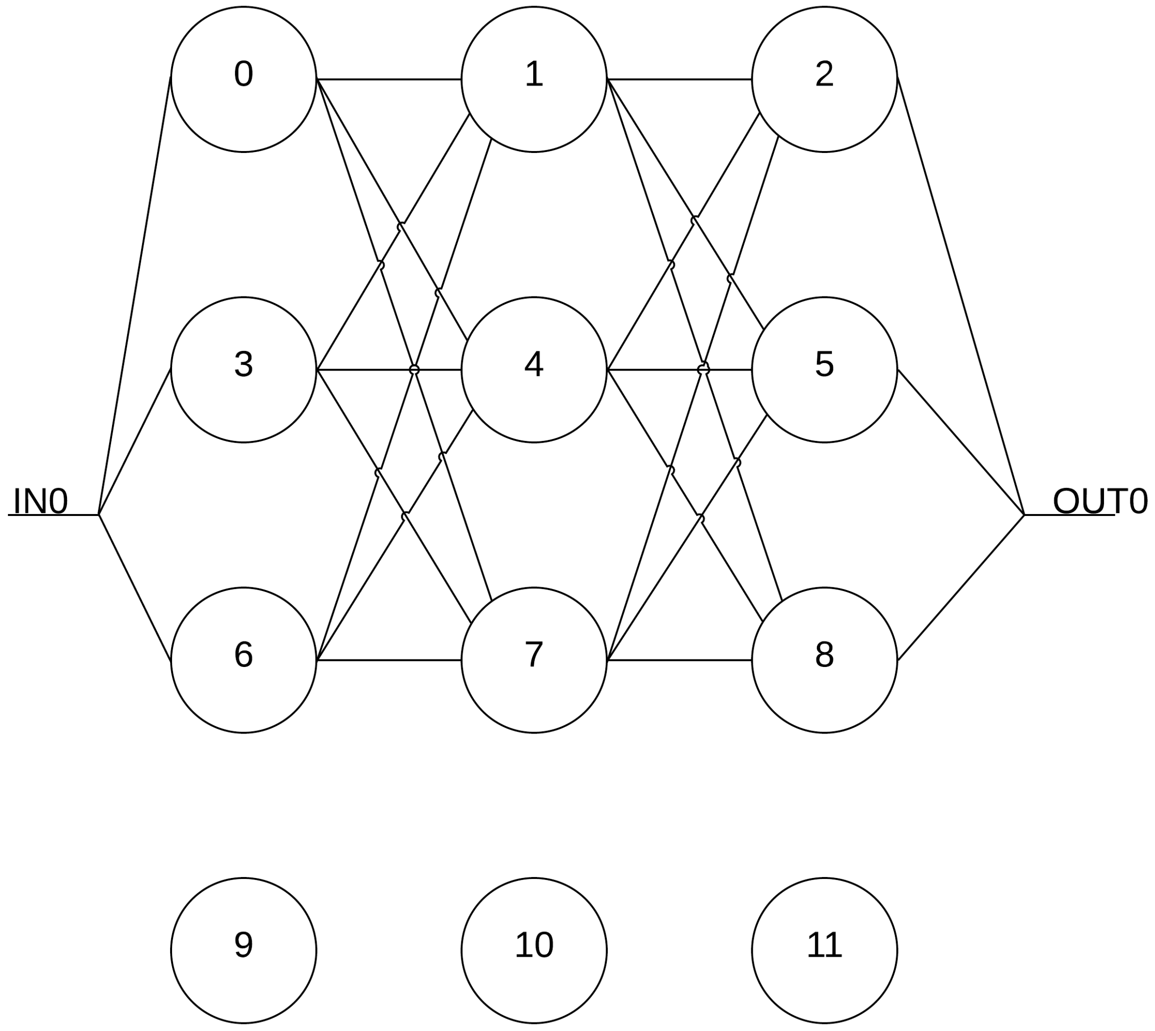
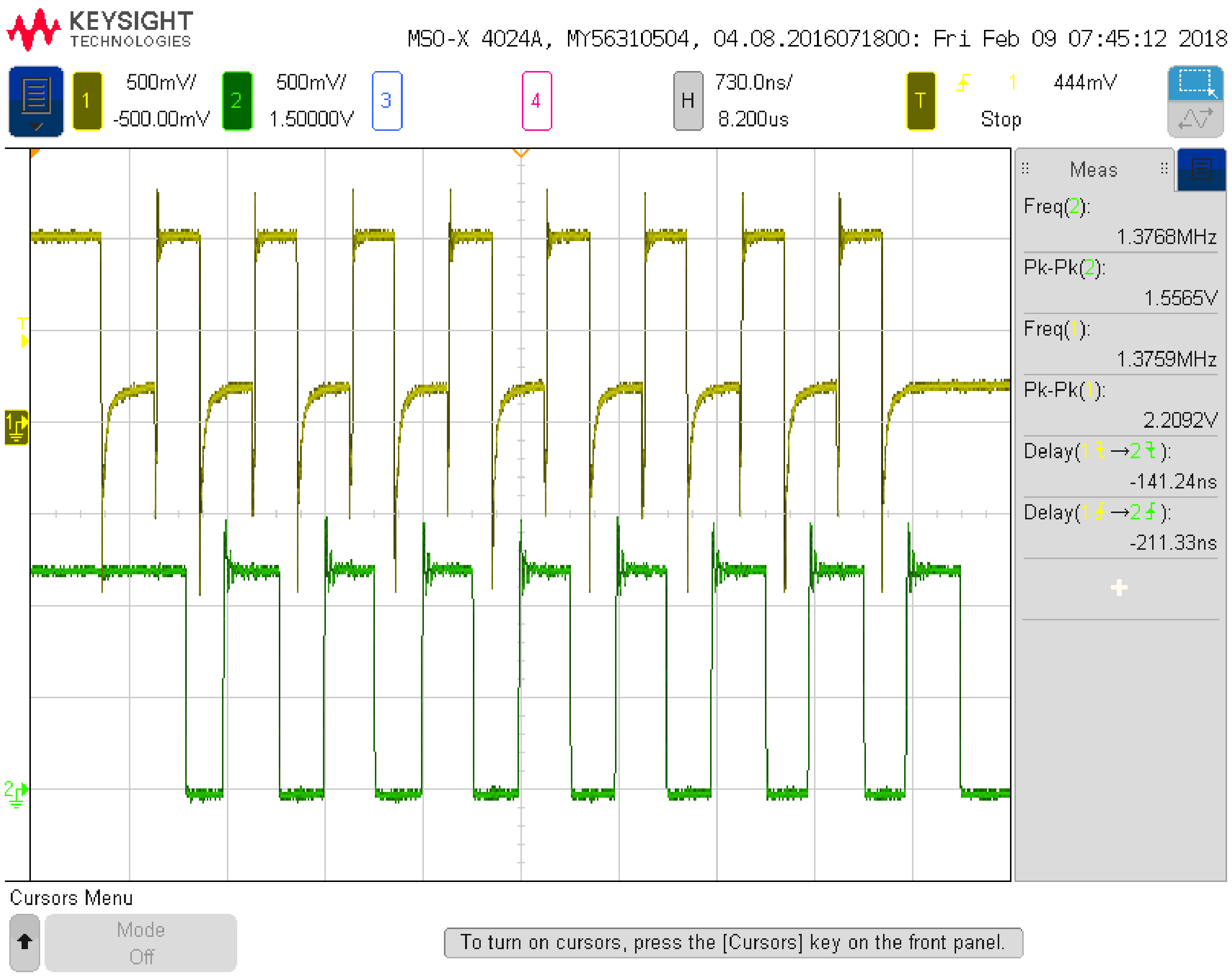
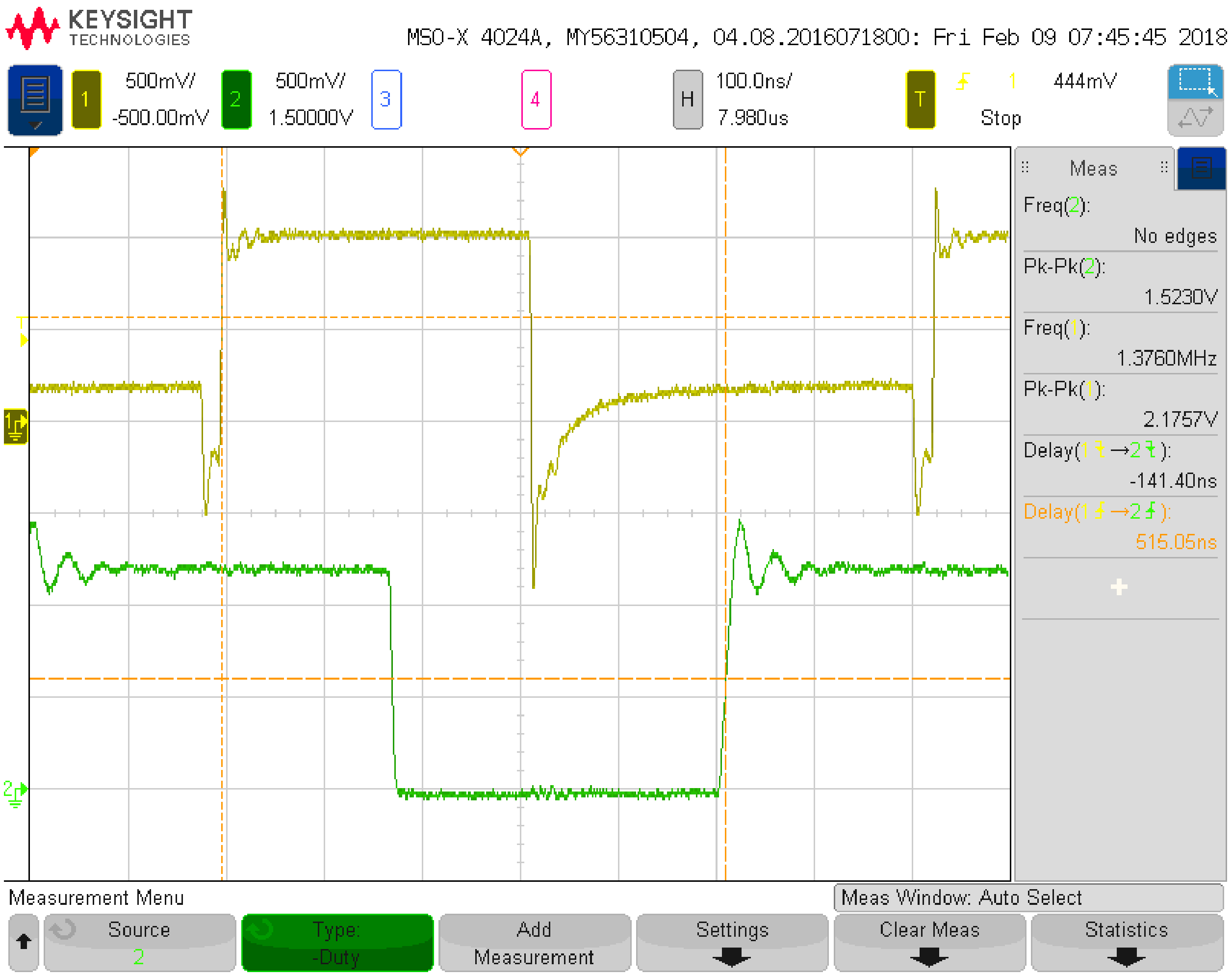


Figure B.13: Fifth MLP Configuration for Measurement Results.



```
MSO-X 4024A, MY56310504, 04.08.2016071800: Fri Feb 09 07:45:12 2018

1  500mV/     2  500mV/    3    4    H  730.0ns/    T  1  444mV
   -500.00mV     1.50000V              8.200us        Stop

Meas
Freq(2):
        1.3768MHz
Pk-Pk(2):
          1.5565V
Freq(1):
        1.3759MHz
Pk-Pk(1):
          2.2092V
Delay(1↑→2↑):
        -141.24ns
Delay(1↓→2↓):
        -211.33ns

Cursors Menu
Mode
Off
To turn on cursors, press the [Cursors] key on the front panel.
```

Figure B.14: Fifth MLP System Configuration Measurement.



MSO-X 4024A, MY56310504, 04.08.2016071800: Fri Feb 09 07:45:45 2018

1 500mV/ -500.00mV
2 500mV/ 1.50000V
H 100.0ns/ 7.980us
T 1 444mV Stop

Freq(2): No edges
Pk-Pk(2): 1.5230V
Freq(1): 1.3760MHz
Pk-Pk(1): 2.1757V
Delay(1→2): -141.40ns
Delay(1→2): 515.05ns

Meas Window: Auto Select

Figure B.15: Fifth MLP System Configuration Measurement for Rising Edge Propagation Delay.

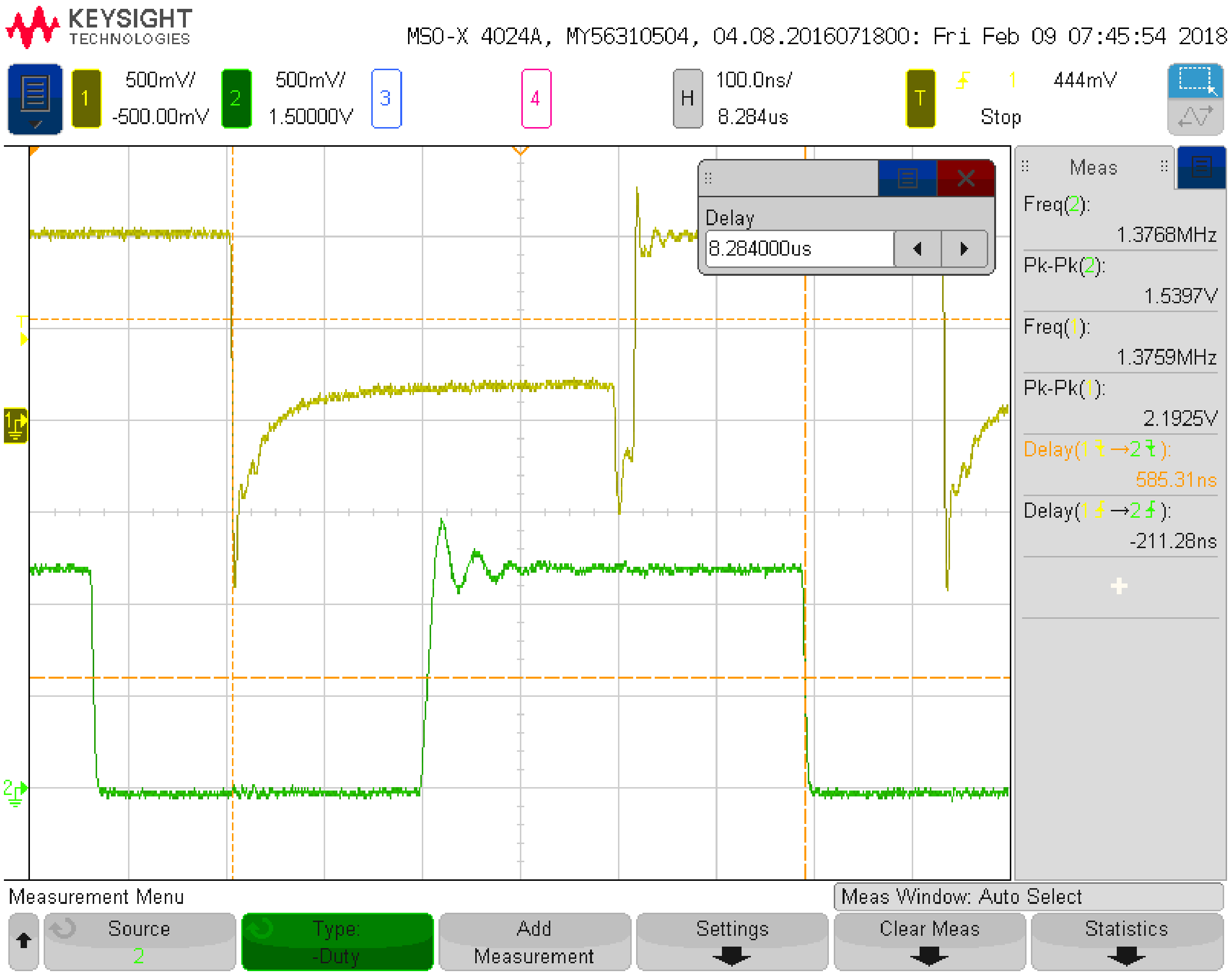


Figure B.16: Fifth MLP System Configuration Measurement for Falling Edge Propagation Delay.

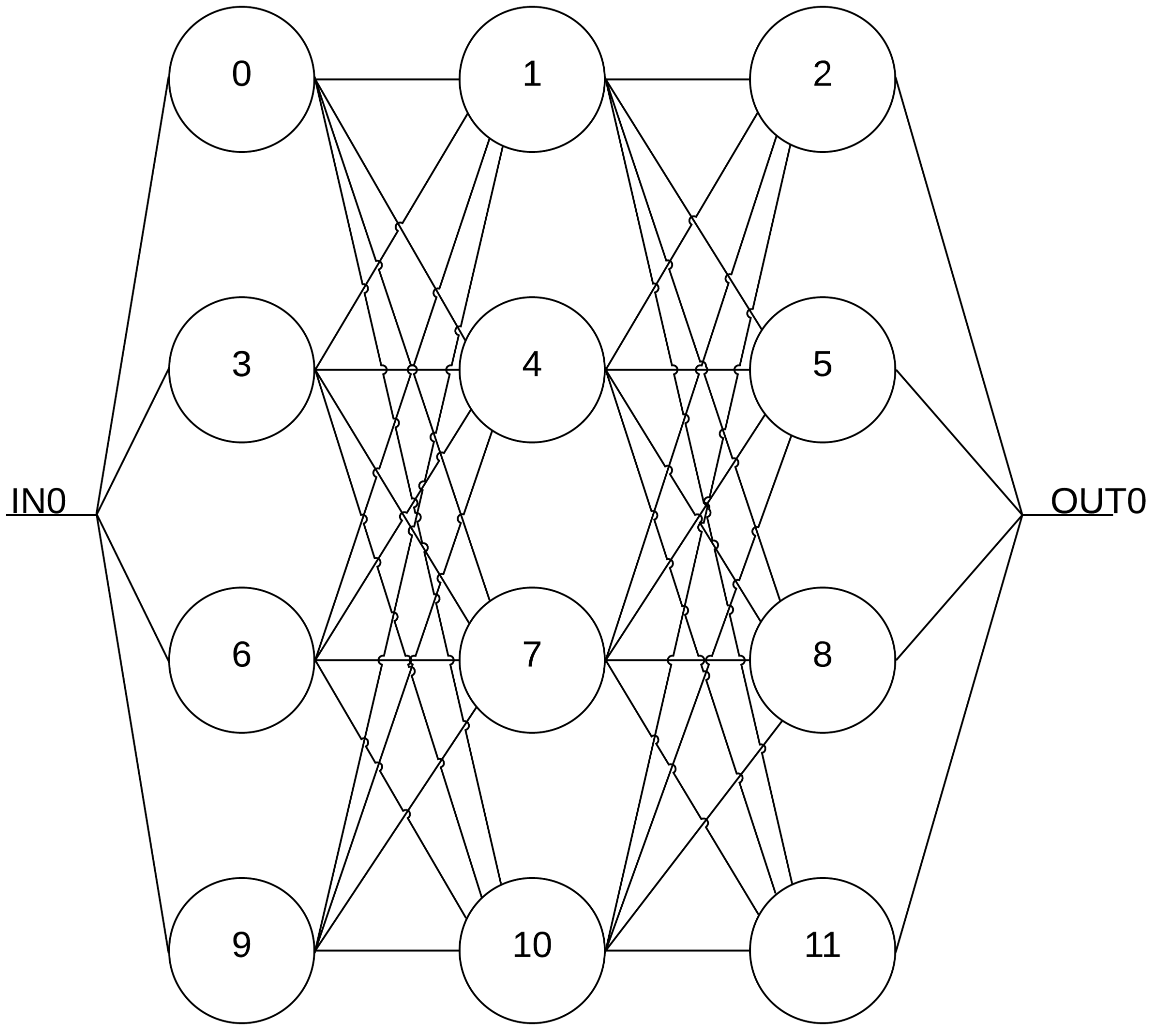
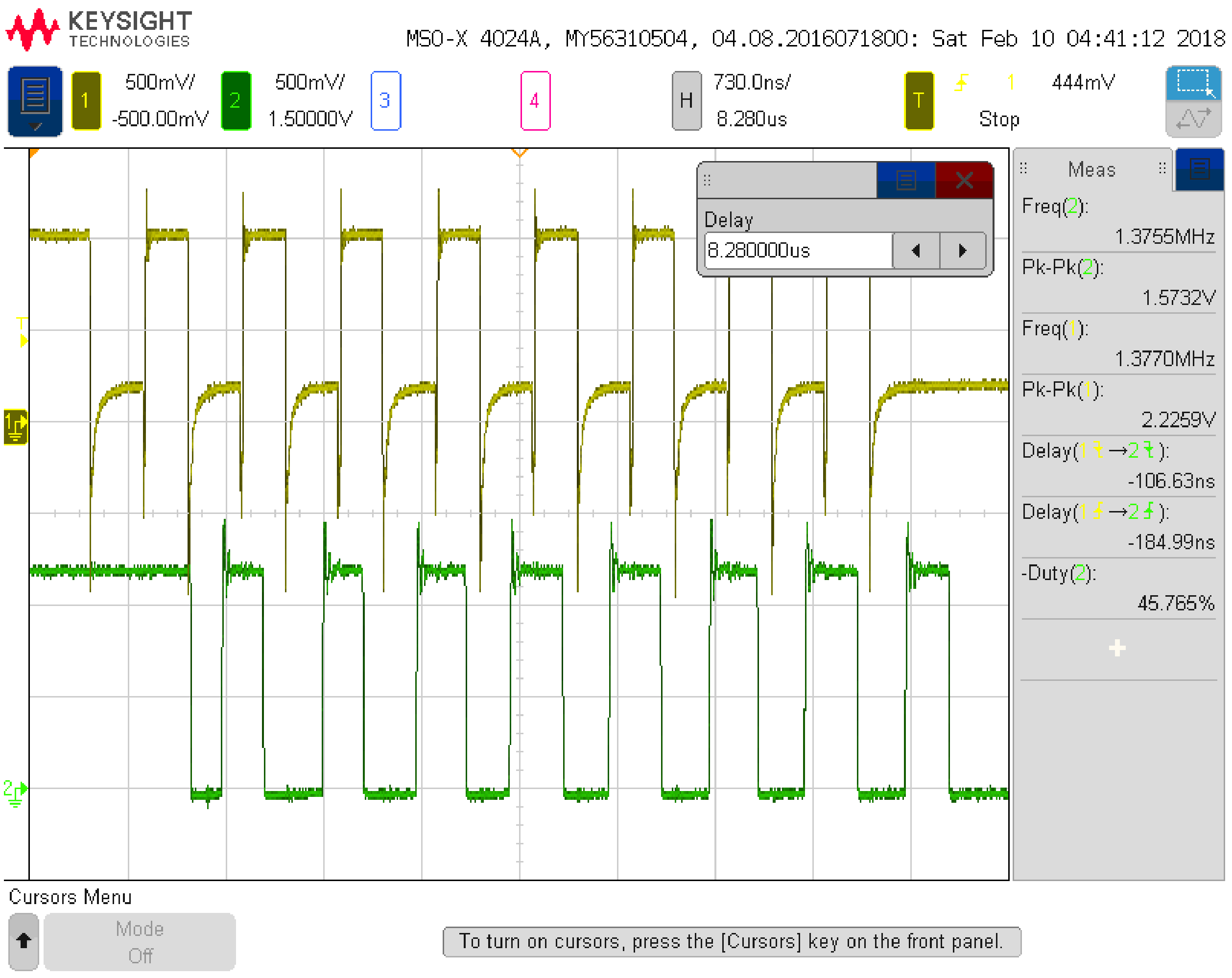


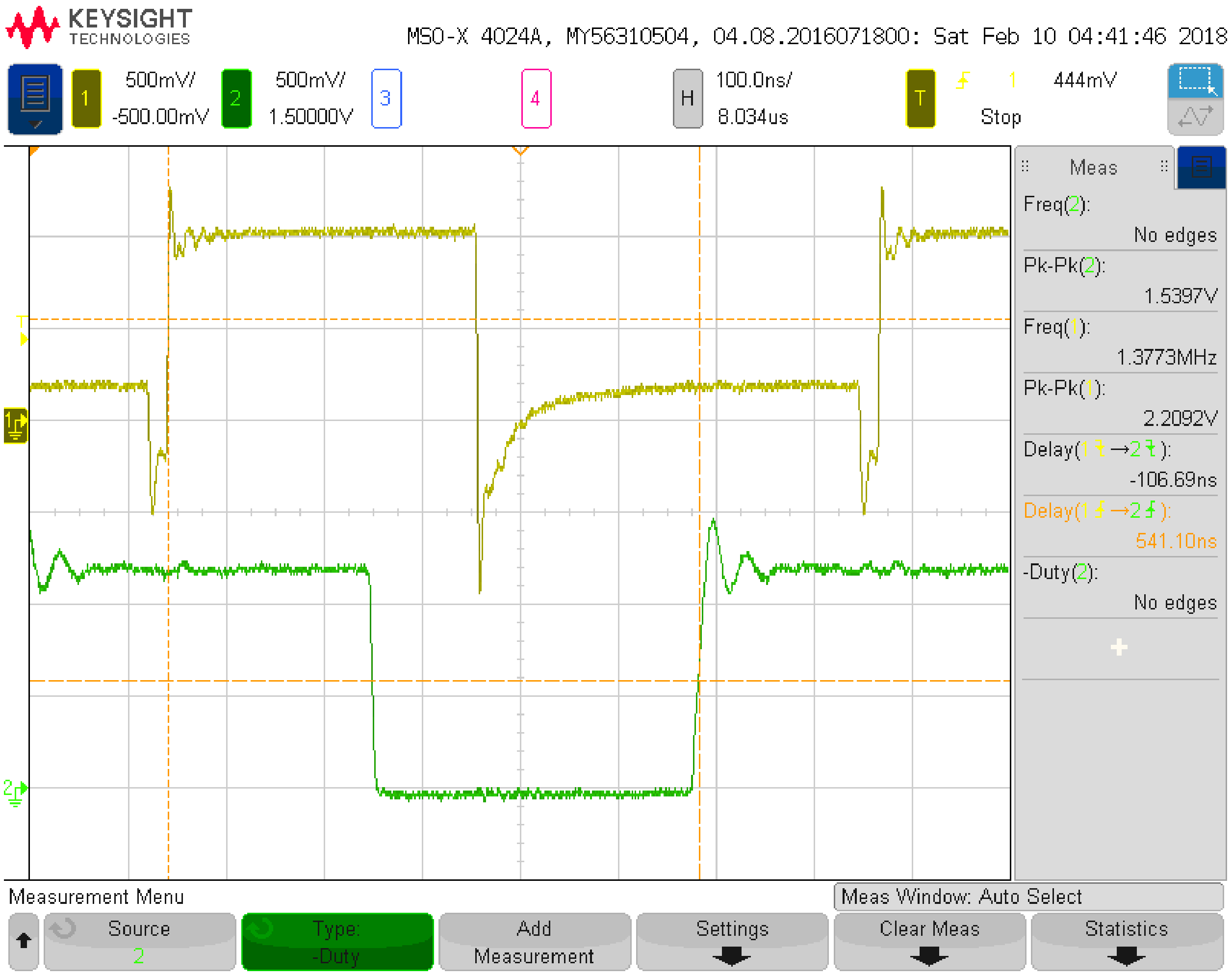
Figure B.17: Sixth MLP Configuration for Measurement Results.



Freq(2): 1.3755MHz
Pk-Pk(2): 1.5732V
Freq(1): 1.3770MHz
Pk-Pk(1): 2.2259V
Delay(1↑→2↑): -106.63ns
Delay(1↓→2↓): -184.99ns
-Duty(2): 45.765%

To turn on cursors, press the [Cursors] key on the front panel.

Figure B.18: Sixth MLP System Configuration Measurement.



Freq(2): No edges
Pk-Pk(2): 1.5397V
Freq(1): 1.3773MHz
Pk-Pk(1): 2.2092V
Delay(1↓→2↓): -106.69ns
Delay(1↑→2↑): 541.10ns
-Duty(2): No edges

Figure B.19: Sixth MLP System Configuration Measurement for Rising Edge Propagation Delay.

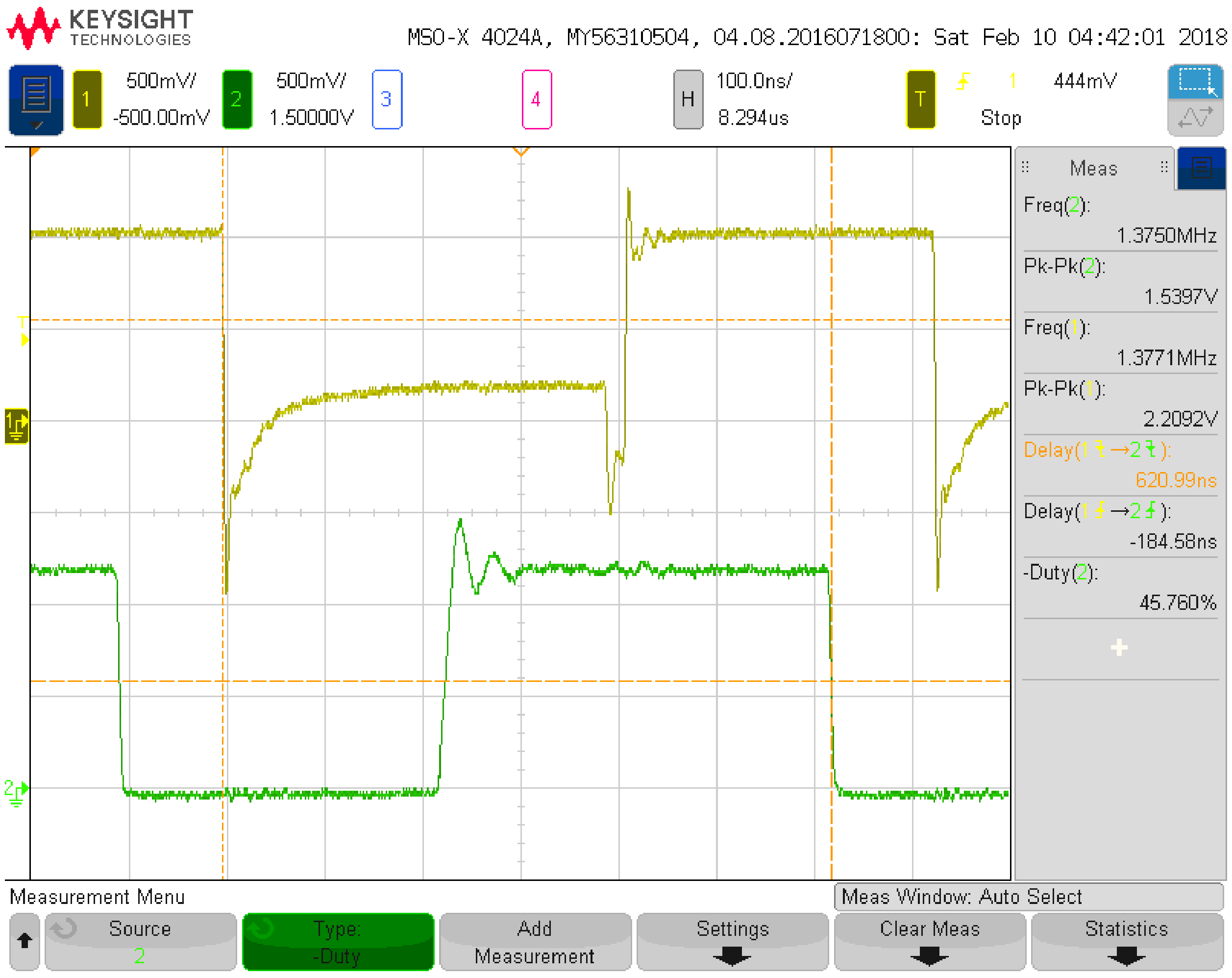


Figure B.20: Sixth MLP System Configuration Measurement for Falling Edge Propagation Delay.

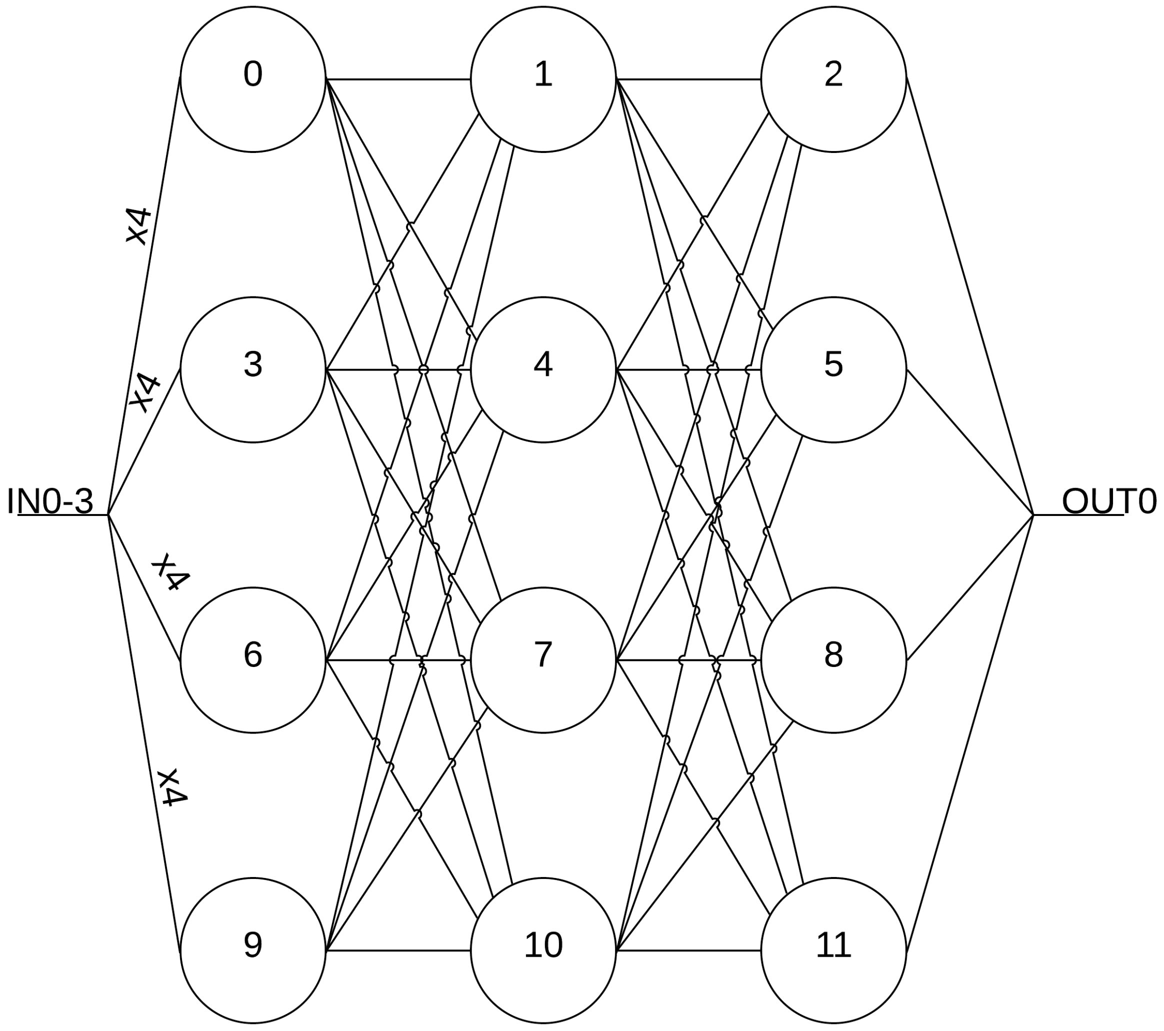
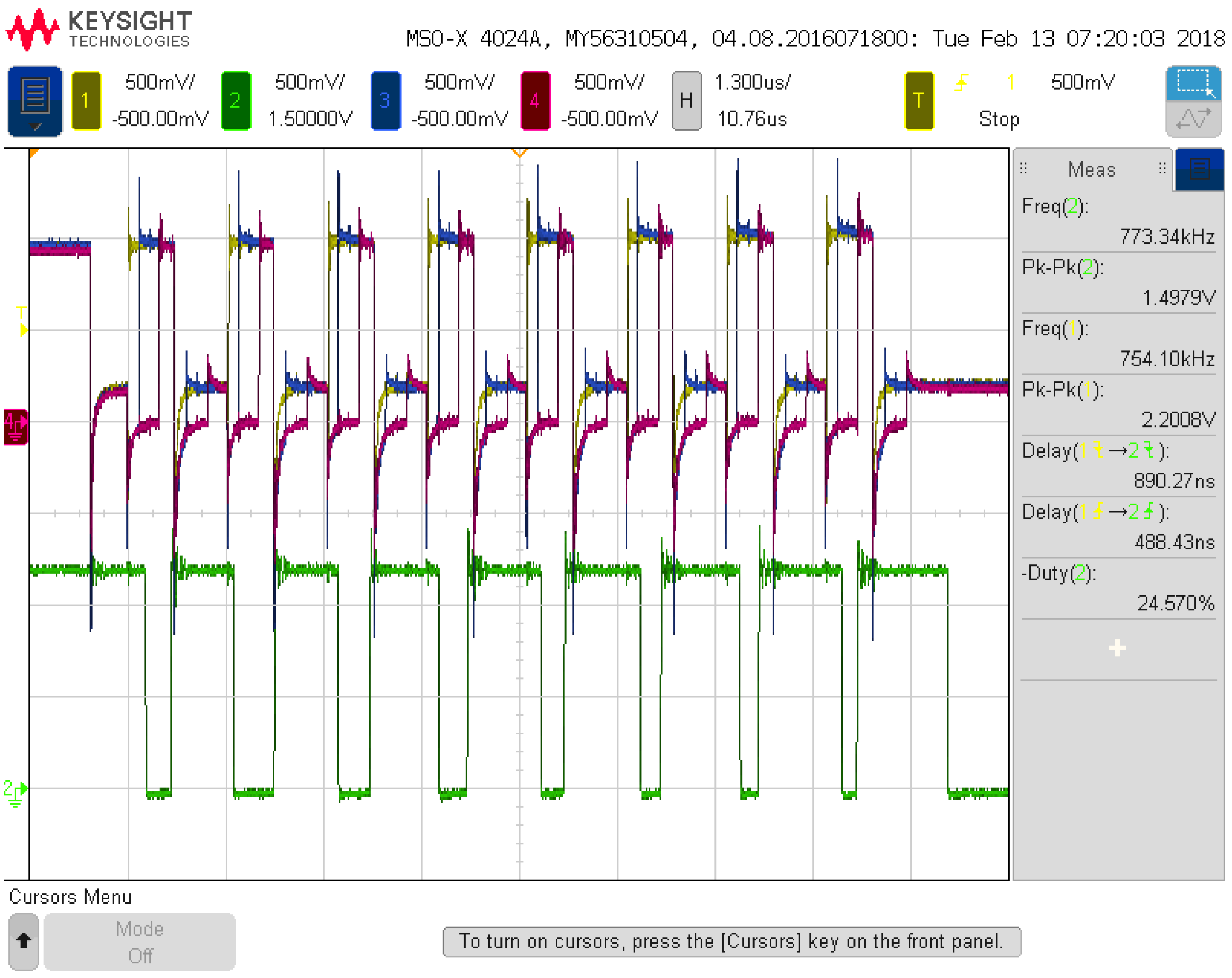


Figure B.21: Seventh MLP Configuration for Measurement Results.



MSO-X 4024A, MY56310504, 04.08.2016071800: Tue Feb 13 07:20:03 2018

500mV/ -500.00mV 500mV/ 1.50000V 500mV/ -500.00mV 500mV/ -500.00mV 1.300us/ 10.76us 1 500mV Stop

Meas

Freq(2): 773.34kHz

Pk-Pk(2): 1.4979V

Freq(1): 754.10kHz

Pk-Pk(1): 2.2008V

Delay(1↑→2↑): 890.27ns

Delay(1↓→2↓): 488.43ns

-Duty(2): 24.570%

To turn on cursors, press the [Cursors] key on the front panel.

Figure B.22: Seventh MLP System Configuration Measurement.

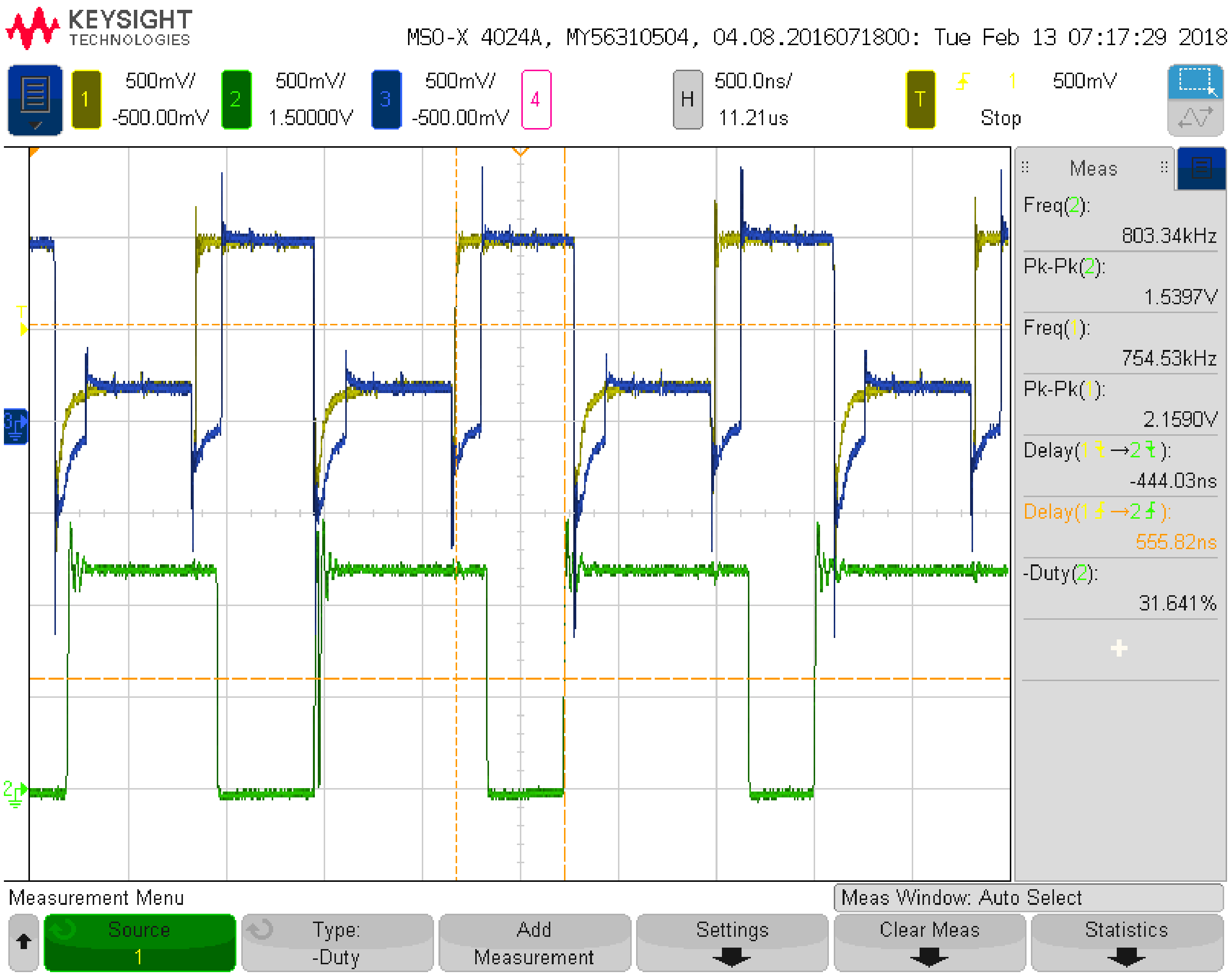
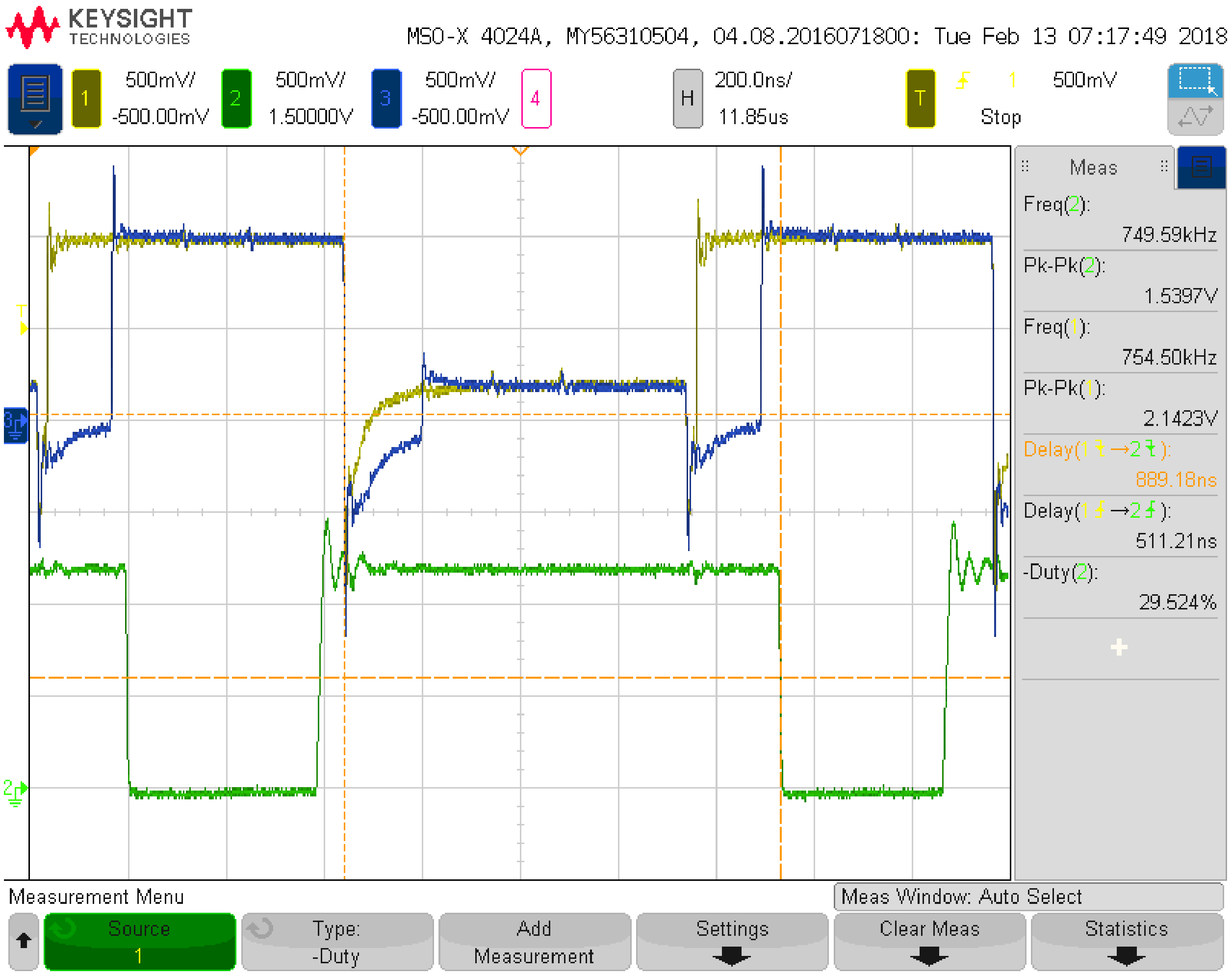


Figure B.23: Seventh MLP System Configuration Measurement for Rising Edge Propagation Delay.



Freq(2): 749.59kHz
Pk-Pk(2): 1.5397V
Freq(1): 754.50kHz
Pk-Pk(1): 2.1423V
Delay(1→2): 889.18ns
Delay(1→2): 511.21ns
-Duty(2): 29.524%

Figure B.24: Seventh MLP System Configuration Measurement for Falling Edge Propagation Delay.

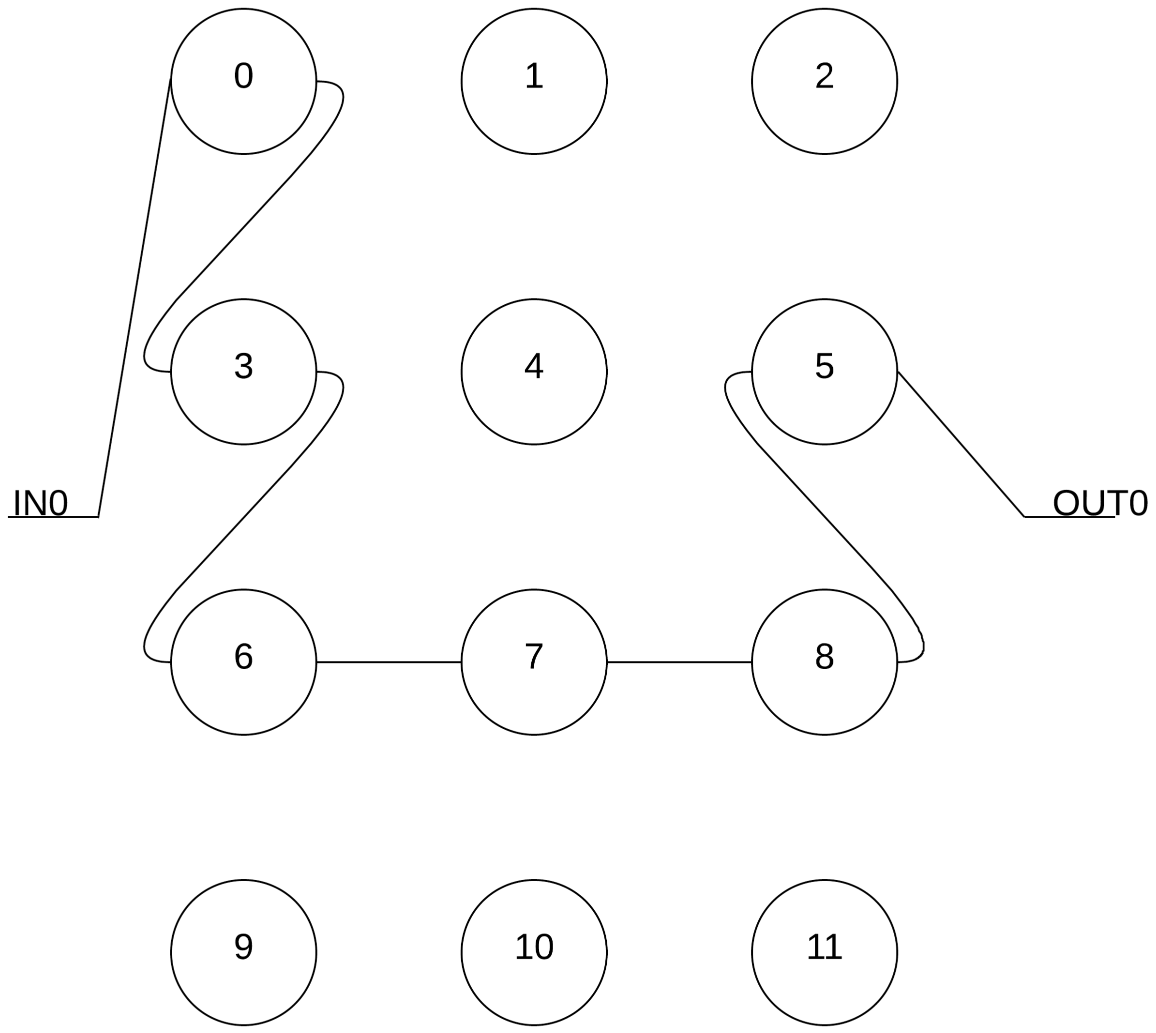
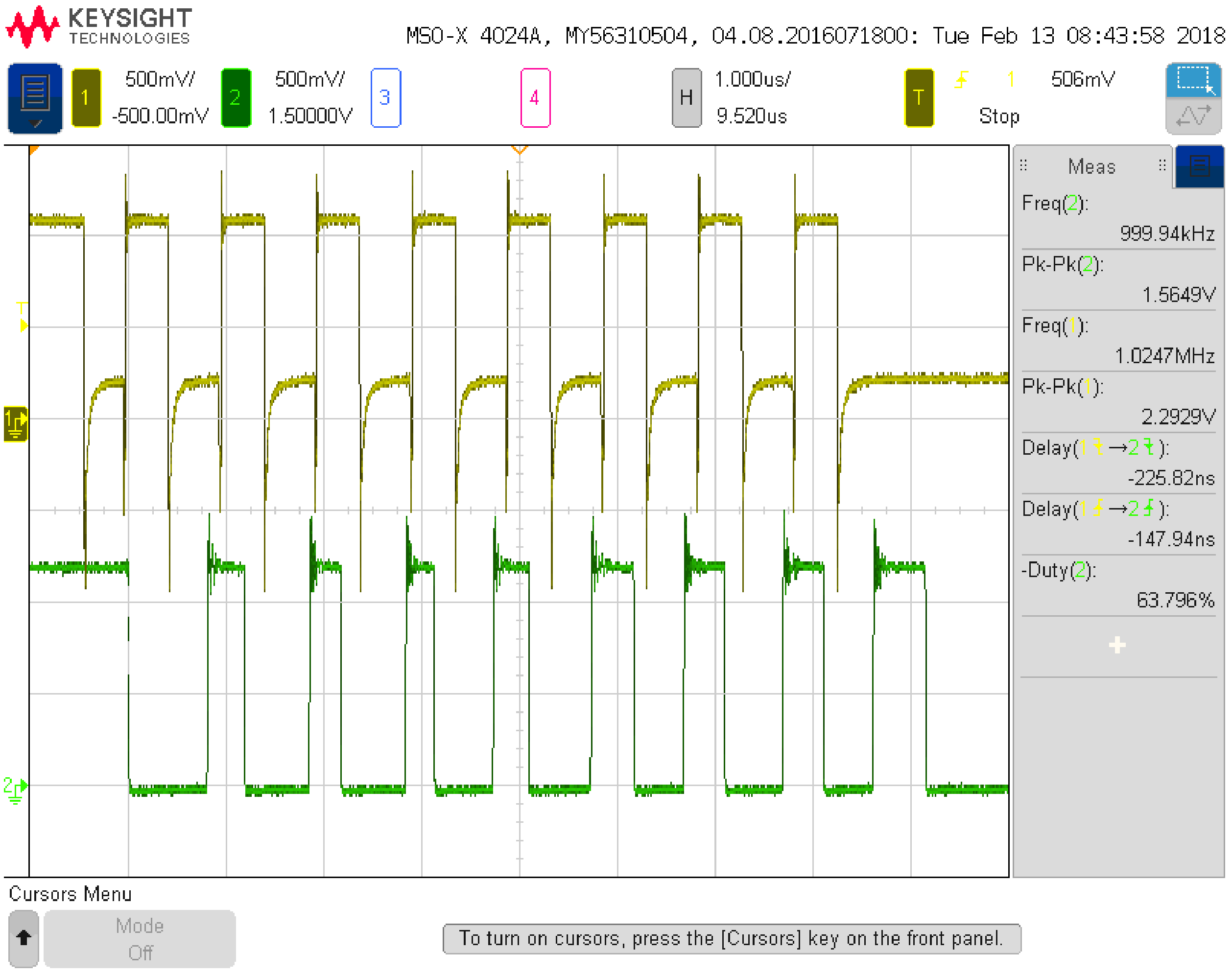


Figure B.25: Eighth MLP Configuration for Measurement Results.



Freq(2): 999.94kHz
Pk-Pk(2): 1.5649V
Freq(1): 1.0247MHz
Pk-Pk(1): 2.2929V
Delay(1↑→2↑): -225.82ns
Delay(1↓→2↓): -147.94ns
-Duty(2): 63.796%

To turn on cursors, press the [Cursors] key on the front panel.

Figure B.26: Eighth MLP System Configuration Measurement.

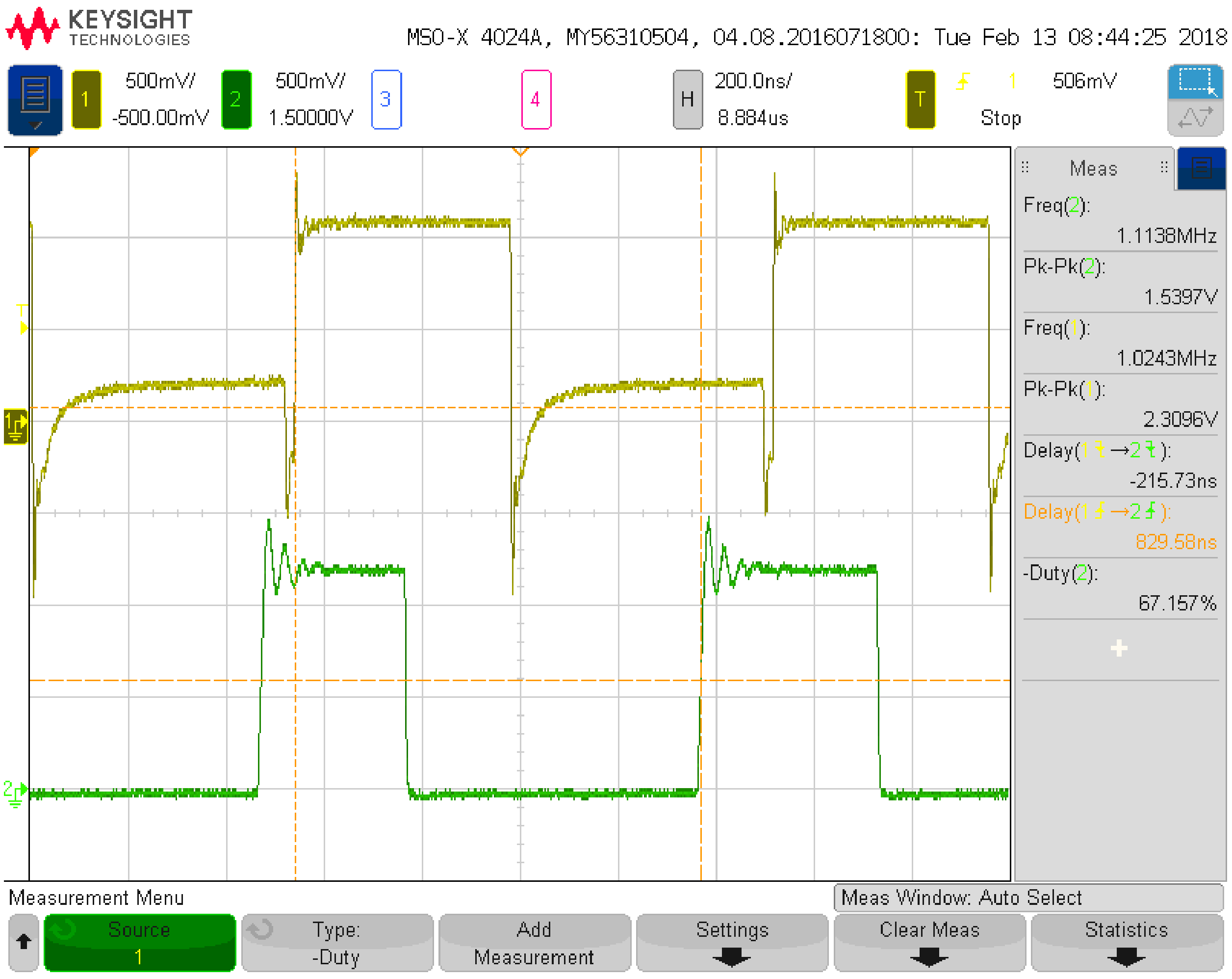
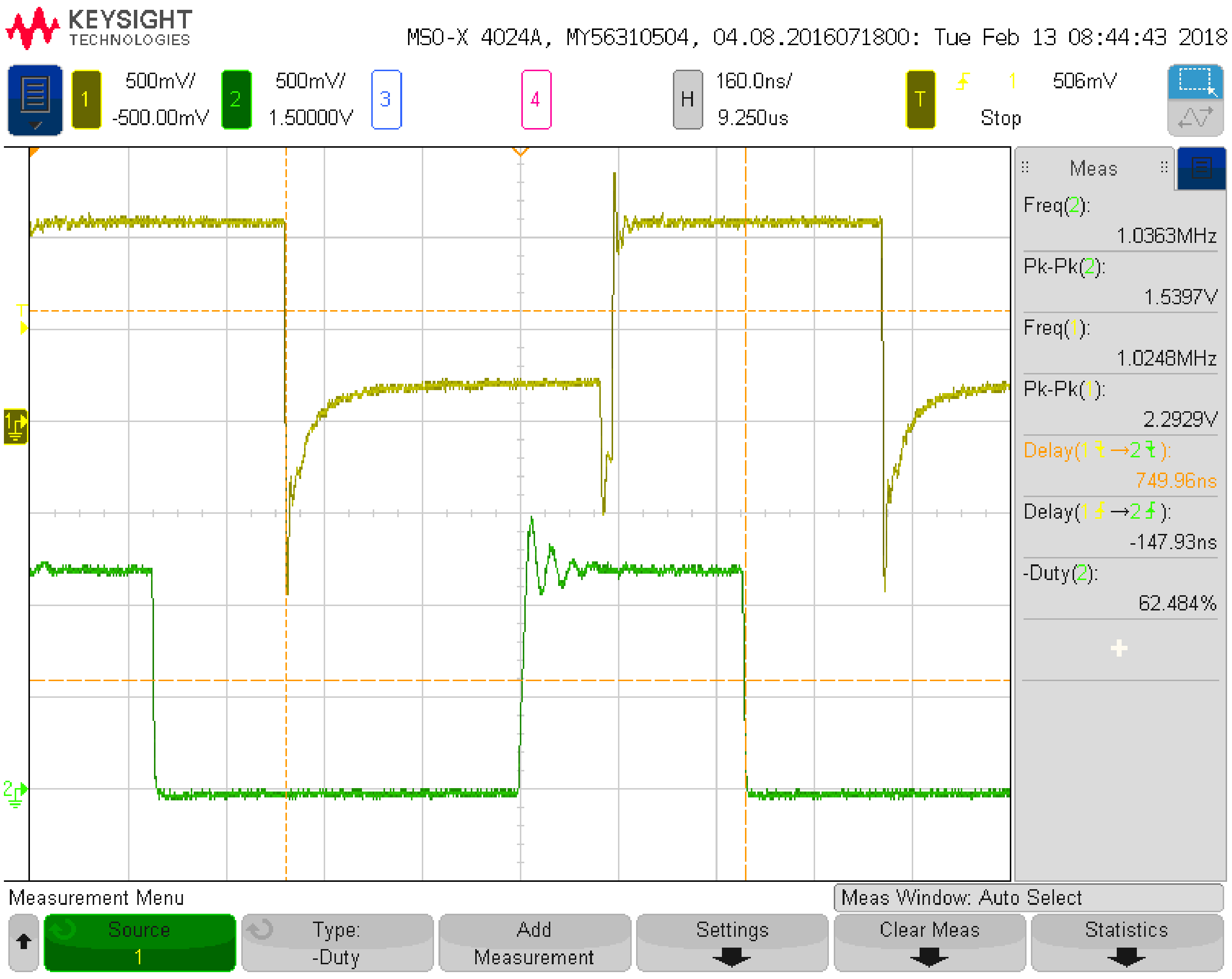


Figure B.27: Eighth MLP System Configuration Measurement for Rising Edge Propagation Delay.


KEYSIGHT TECHNOLOGIES

MSO-X 4024A, MY56310504, 04.08.2016071800: Tue Feb 13 08:44:43 2018

Freq(2): 1.0363MHz
Pk-Pk(2): 1.5397V
Freq(1): 1.0248MHz
Pk-Pk(1): 2.2929V
Delay(1↑→2↑): 749.96ns
Delay(1↓→2↓): -147.93ns
-Duty(2): 62.484%

Measurement Menu

Meas Window: Auto Select

Figure B.28: Eighth MLP System Configuration Measurement for Falling Edge Propagation Delay.

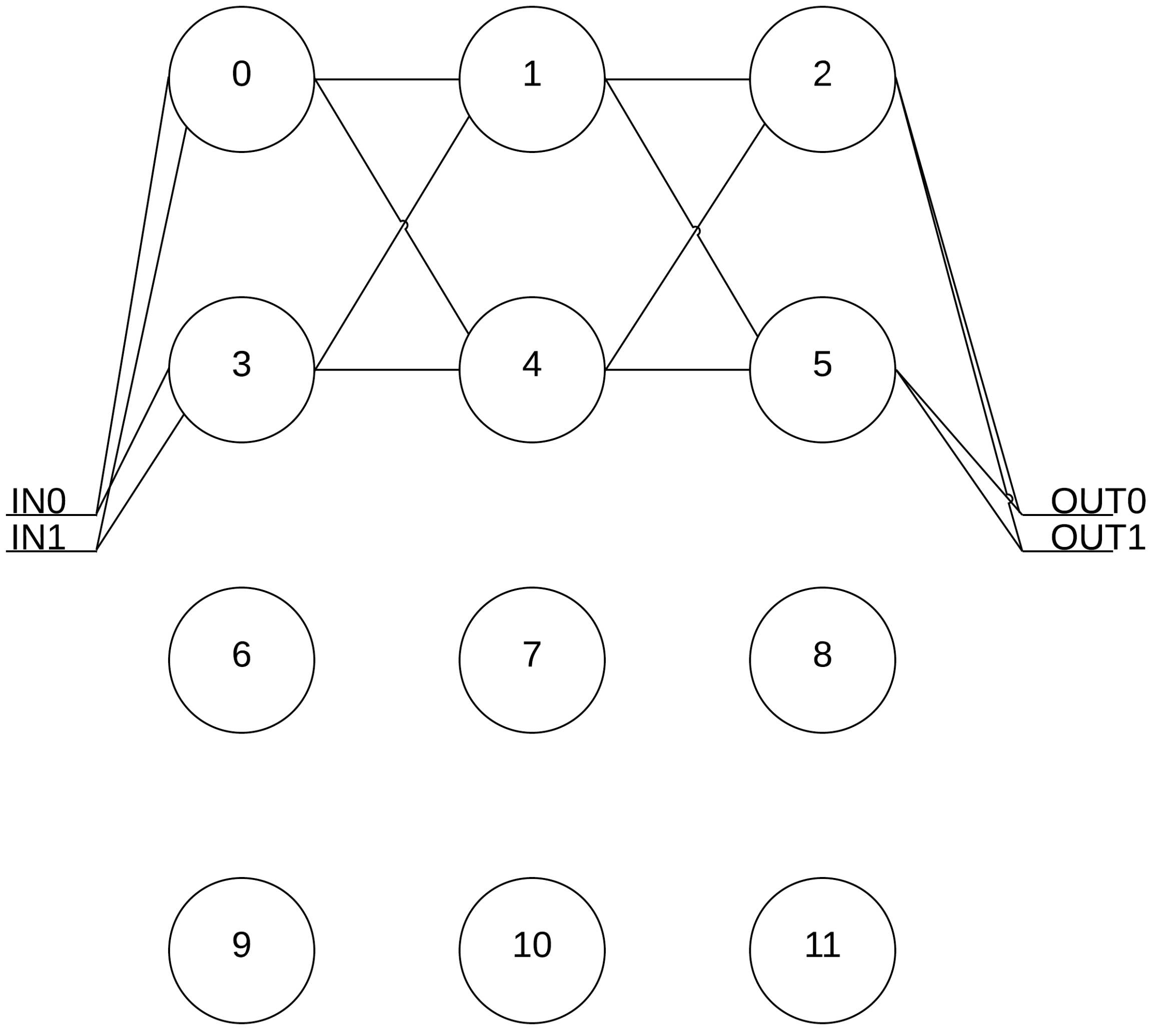
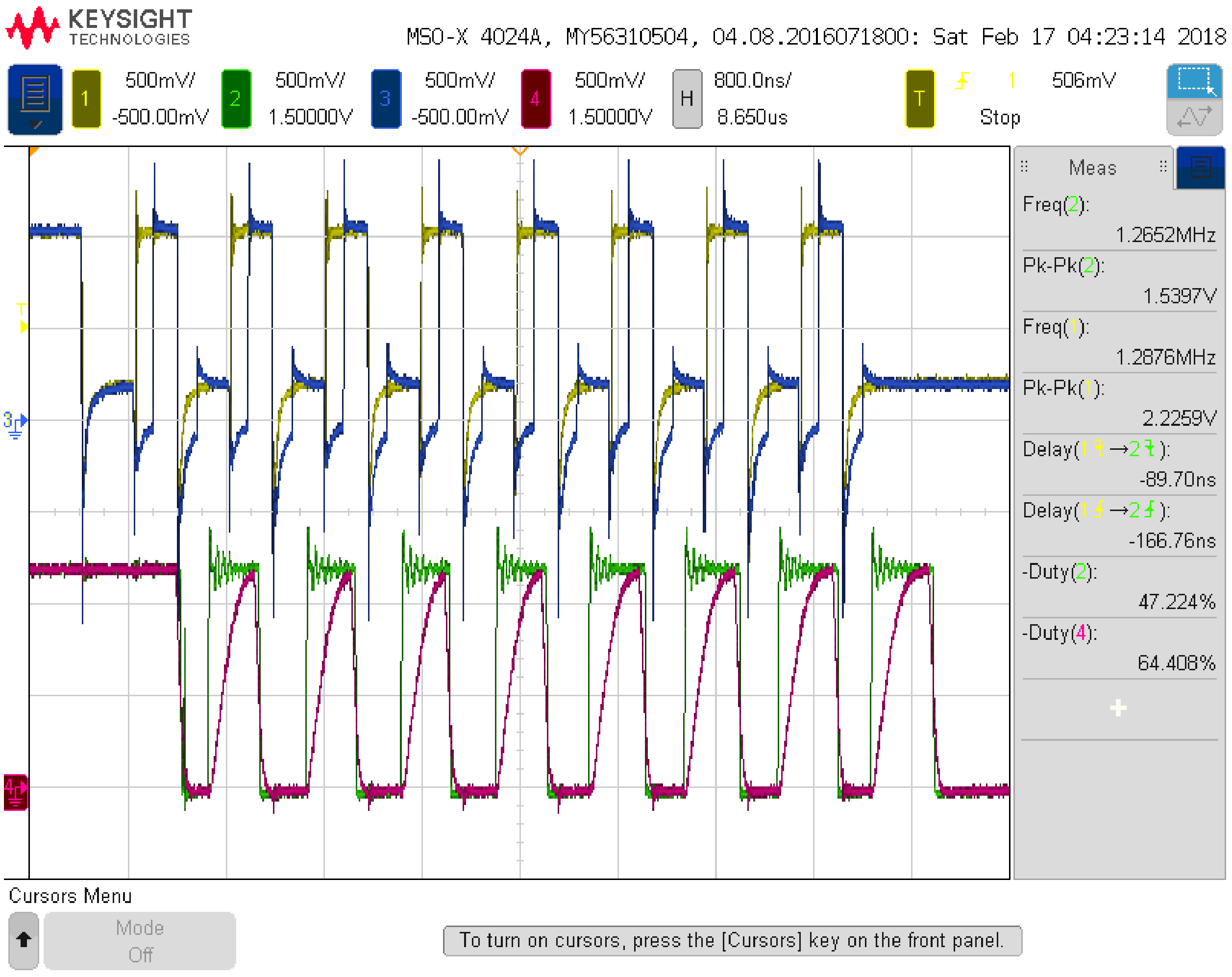


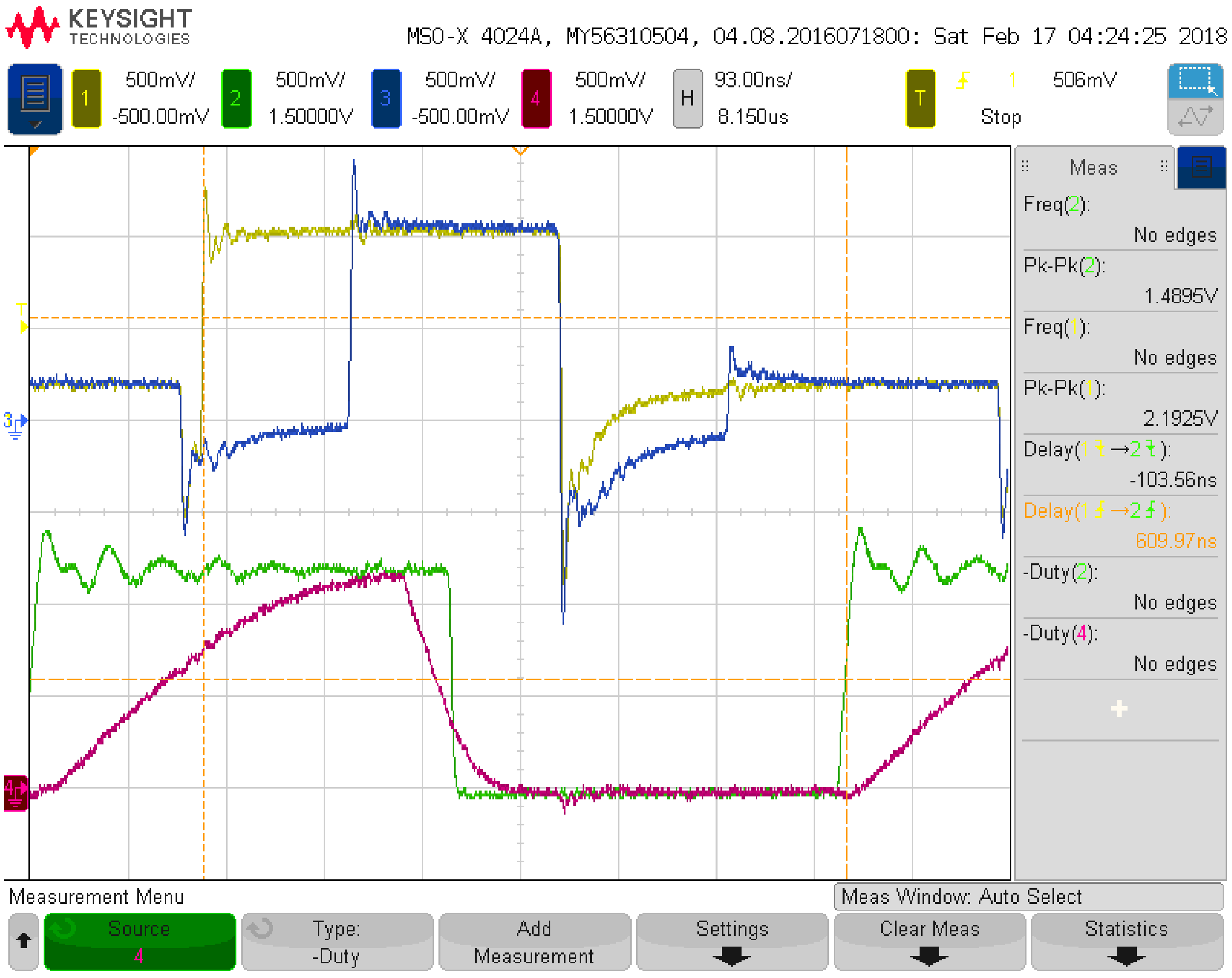
Figure B.29: Ninth MLP Configuration for Measurement Results.



Freq(2): 1.2652MHz
Pk-Pk(2): 1.5397V
Freq(1): 1.2876MHz
Pk-Pk(1): 2.2259V
Delay(1↑→2↑): -89.70ns
Delay(1↓→2↓): -166.76ns
-Duty(2): 47.224%
-Duty(4): 64.408%

To turn on cursors, press the [Cursors] key on the front panel.

Figure B.30: Ninth MLP System Configuration Measurement.



MSO-X 4024A, MY56310504, 04.08.2016071800: Sat Feb 17 04:24:25 2018

500mV/ -500.00mV 500mV/ 1.50000V 500mV/ -500.00mV 500mV/ 1.50000V 93.00ns/ 8.150us 1 506mV

Freq(2): No edges
Pk-Pk(2): 1.4895V
Freq(1): No edges
Pk-Pk(1): 2.1925V
Delay(1 ↑ →2 ↑): -103.56ns
Delay(1 ↓ →2 ↓): 609.97ns
-Duty(2): No edges
-Duty(4): No edges

Meas Window: Auto Select

Figure B.31: Ninth MLP System Configuration Measurement for Rising Edge Propagation Delay.

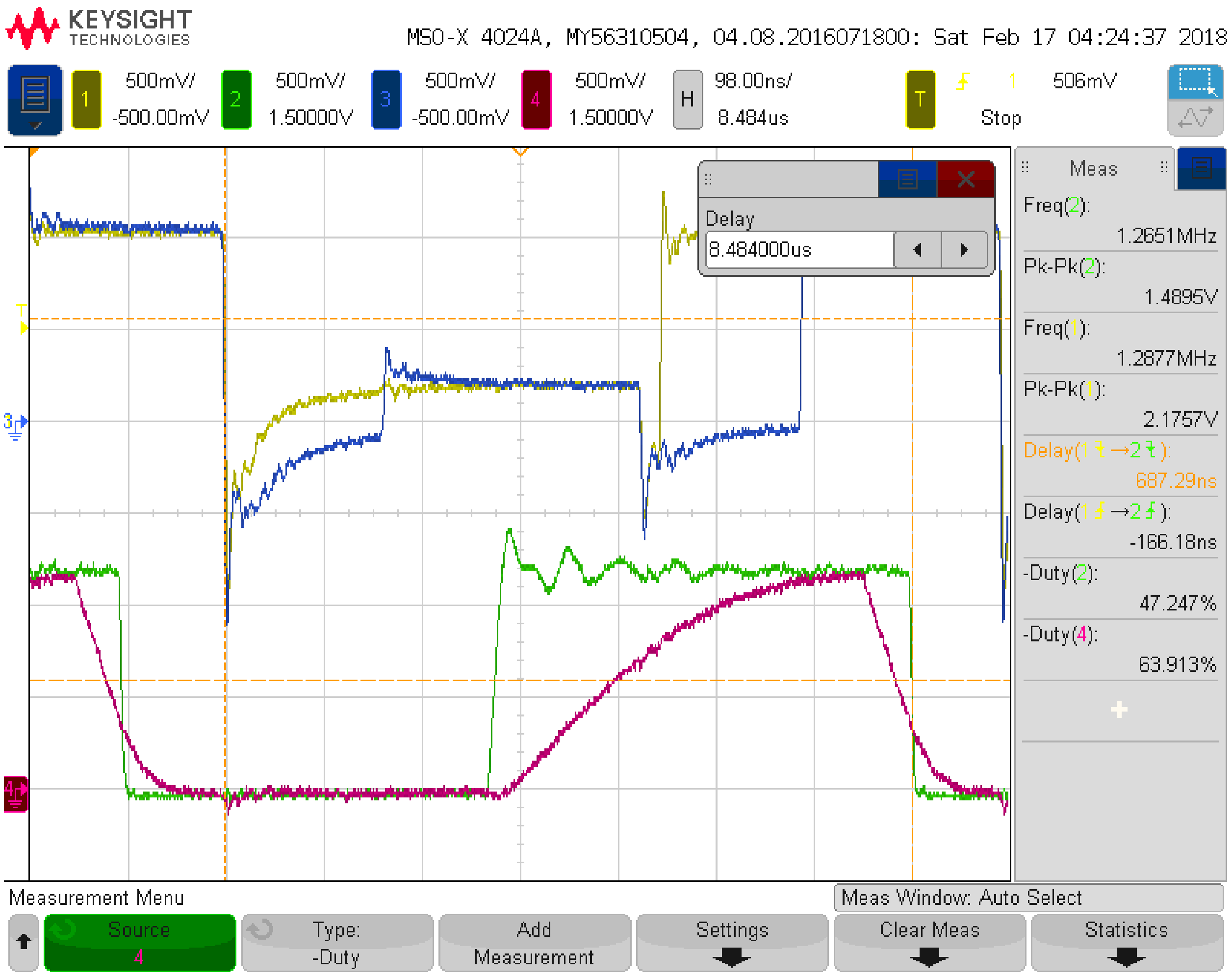


Figure B.32: Ninth MLP System Configuration Measurement for Falling Edge Propagation Delay.

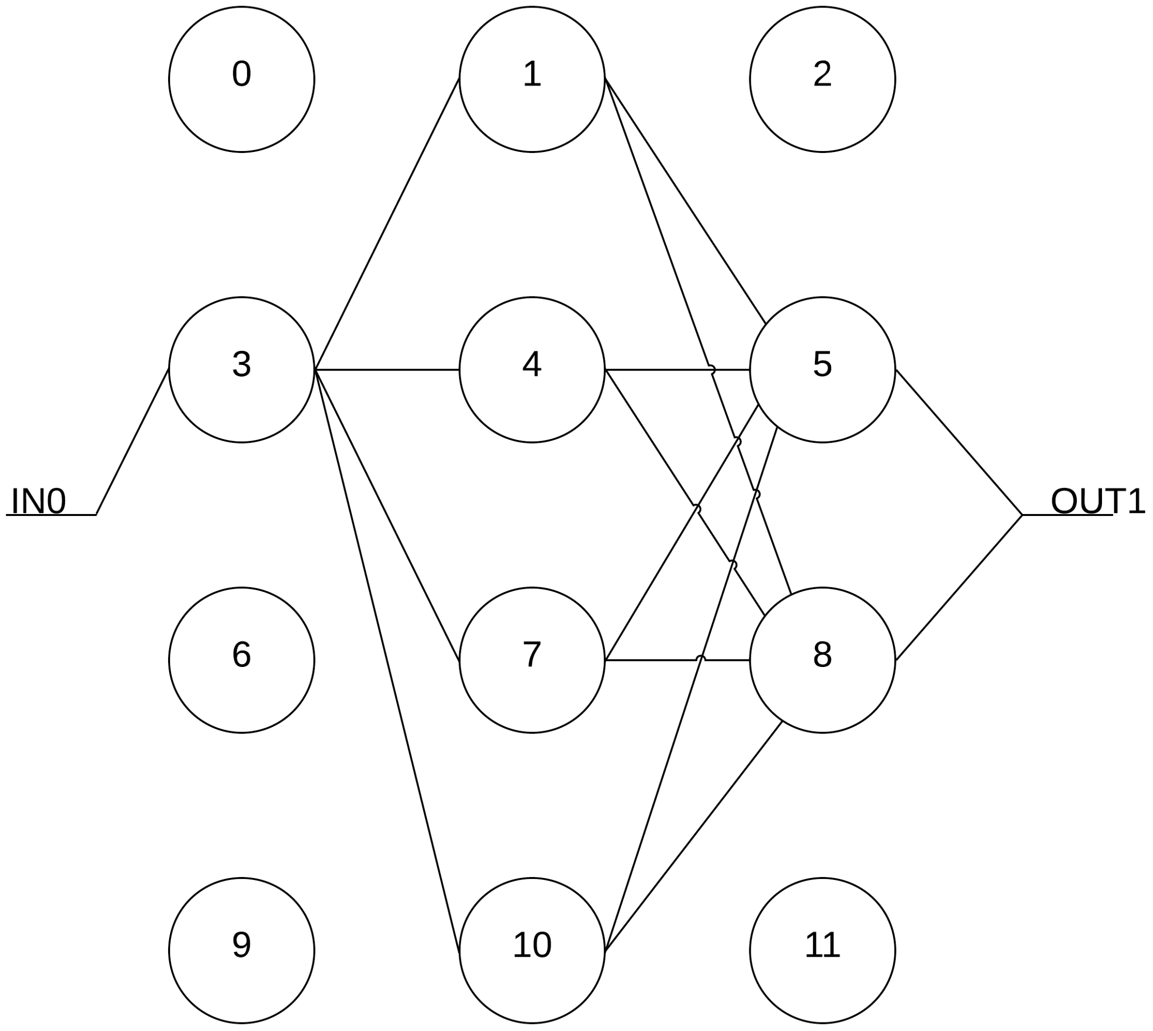
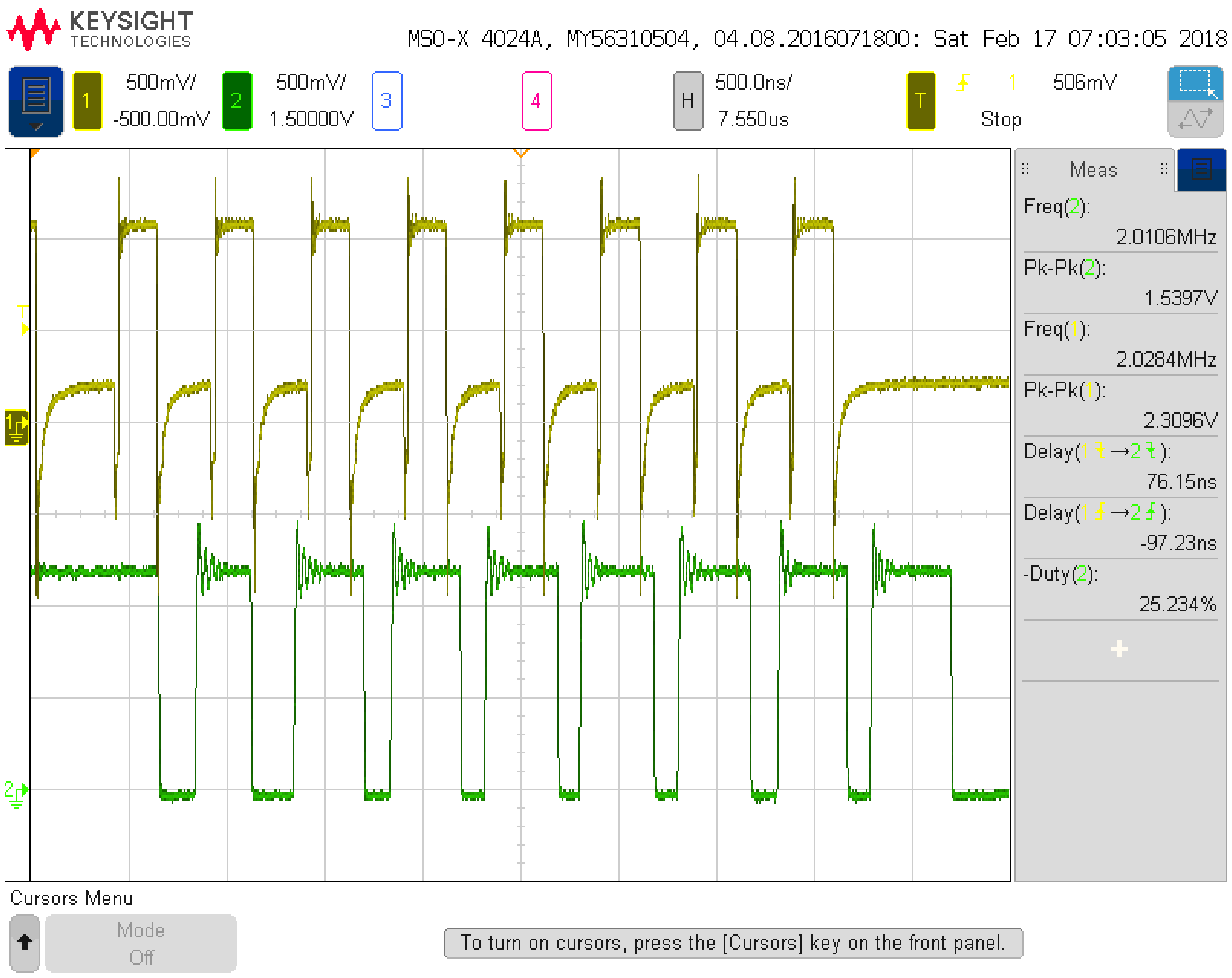


Figure B.33: Tenth MLP Configuration for Measurement Results.


MSO-X 4024A, MY56310504, 04.08.2016071800: Sat Feb 17 07:03:05 2018

1 500mV/ -500.00mV
2 500mV/ 1.50000V
H 500.0ns/ 7.550us
T 1 506mV Stop

Meas
Freq(2): 2.0106MHz
Pk-Pk(2): 1.5397V
Freq(1): 2.0284MHz
Pk-Pk(1): 2.3096V
Delay(1 → 2): 76.15ns
Delay(1 → 2): -97.23ns
-Duty(2): 25.234%

Cursors Menu
Mode Off
To turn on cursors, press the [Cursors] key on the front panel.

Figure B.34: Tenth MLP System Configuration Measurement.

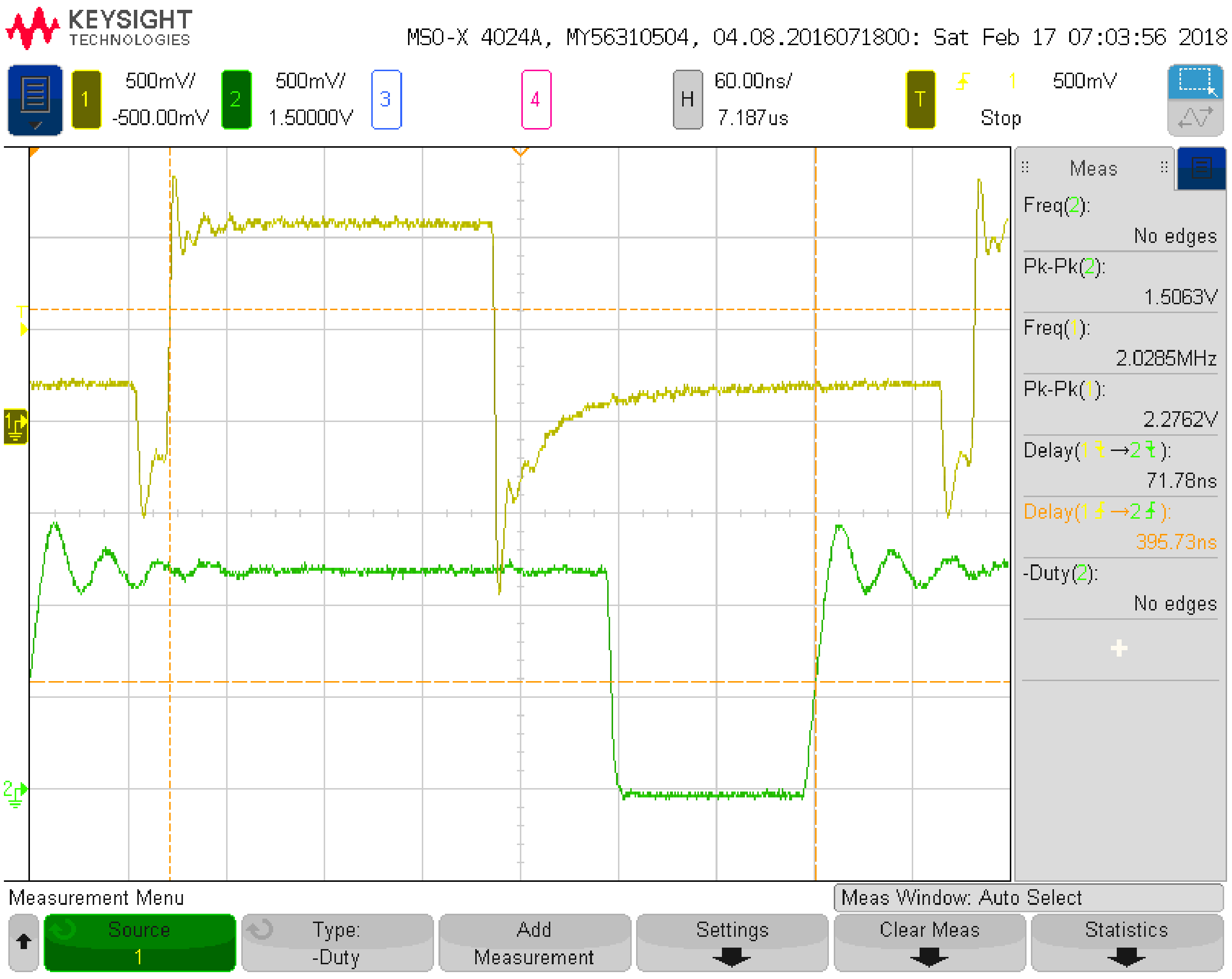


Figure B.35: Tenth MLP System Configuration Measurement for Rising Edge Propagation Delay.

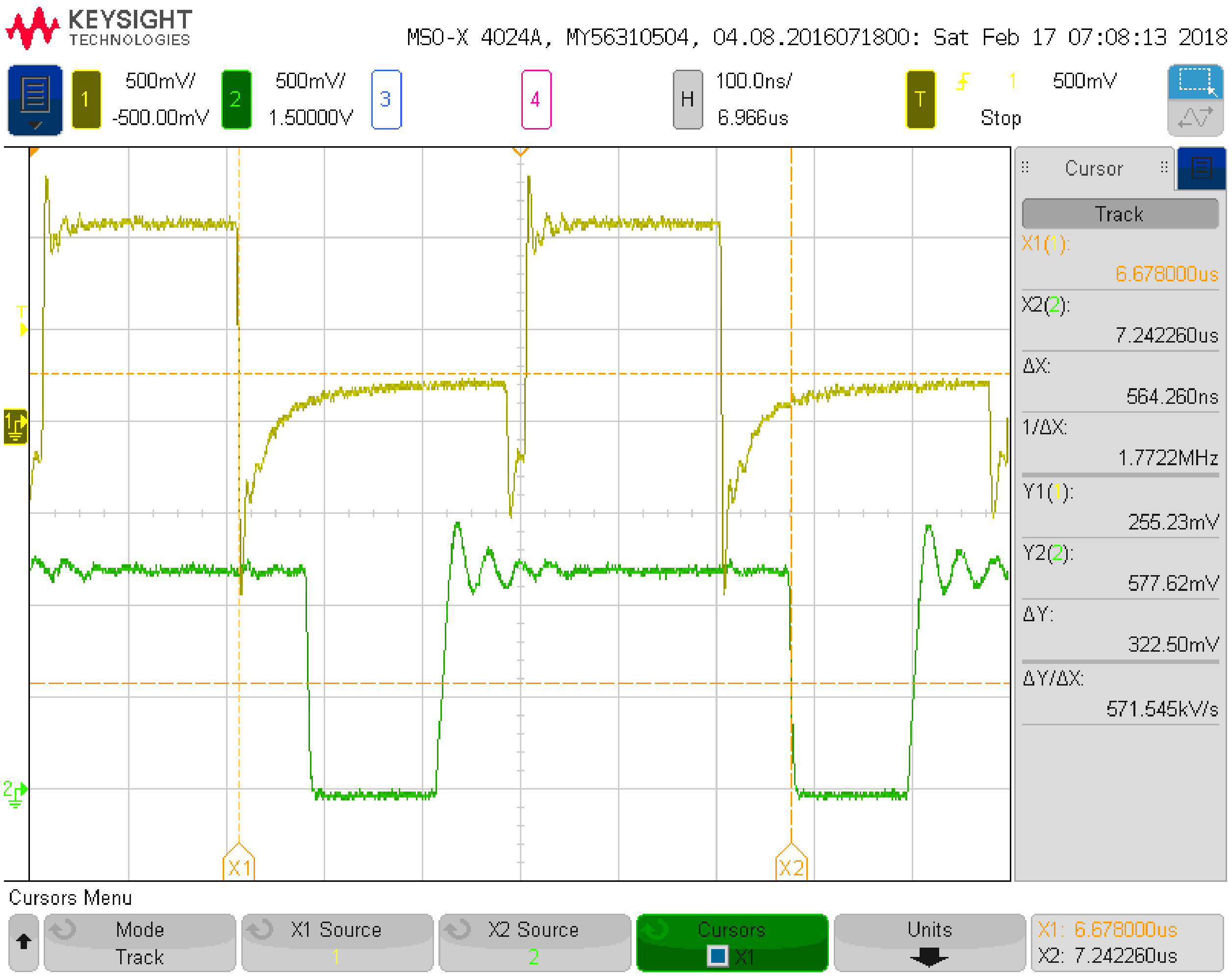


Figure B.36: Tenth MLP System Configuration Measurement for Falling Edge Propagation Delay.

Vita

Jeffery Dix was born in Bristol, TN, to the parents of John and Lisa Dix. He is third of four siblings: John, Matthew, and Kim. He attended Bearden High School in Knoxville, TN. After graduation, he continued in Knoxville at the University of Tennessee, Knoxville, where he studied Electrical Engineering. Jeffery completed a co-op rotation with ADTRAN, INC in Huntsville, AL, as well as interned at Duke Progress Energy in Southport, NC. After obtaining a Bachelors of Science degree at the University of Tennessee in December 2013 in Electrical Engineering, his professors inspired him to continue his education. He accepted the Chancellor's Fellowship at the University of Tennessee, Knoxville, in the Electrical Engineering program and in the Integrated Circuits and Systems Lab. Jeff graduated with a Masters of Science degree in Electrical Engineering in May 2015 while pursuing his Doctor of Philosophy degree. During his graduate career, Jeff interned at the Electric Power Research Institute from May 2015 to August 2016. Jeff graduated with a Doctor of Philosophy degree in Electrical Engineering in May 2018.