Representations and Circuits for Time Based Computation

Khaldoon Mhaidat

B.S., Electrical and Computer Engineering, Jordan University of Science & Technology (1999)

M.S., Electrical and Computer Engineering, Oregon State University (2002)

A dissertation submitted to the faculty of
OGI School of Science & Engineering at
Oregon Health & Science University
in partial fulfillment of the
requirements for the degree
Doctor of Philosophy
in
Electrical and Computer Engineering

March 2006

The dissertation “Representations and Circuits for Time Based Computation” by Khaldoon Mhaidat has been examined and approved by the following Examination Committee:

Dr. Marwan Jabri

Ph.D. Advisor

Gordon and Betty Moore Chair Professor of Electrical and Computer Engineering

OGI School of Science & Engineering at Oregon Health & Science University

---

Dr. Dan Hammerstrom

Professor of Electrical and Computer Engineering and Associate Dean for Research

Maseeh College of Engineering & Computer Science at Portland State University

---

Ali Muhtaroglu

Mobile Platform Group, Intel Corporation

John Lynch

Instructor, Department of Computer Science and Electrical Engineering

OGI School of Science & Engineering at Oregon Health & Science University

Acknowledgments

First and most, I thank and praise God who has created me and bestowed countless blessings upon me and all his creation. To him I dedicate all of this work and to him I pray it will benefit all his creation in a good way.

I thank Dr. Marwan Jabri for giving me the great opportunity to work with him and gain from his long and rich experience. Dr. Marwan's supervision of my Ph.D. research work and his reviews, comments, and feedback on it have been invaluable.

I also thank Dr. Dan Hammerstrom, Ali Muhtaroglu, John Lynch, and Dr. Todd Leen for their invaluable reviews, comments, and feedback on this work.

I also thank Lava Raganathan for his help with chip design tools and John Hunt for his help with chip testing.

I also thank my lovely wife and daughter, Tamara and Huda, for their help, support, and patience during my Ph.D. years.

I also thank my dear mother, father, and family, who have always wanted me to be excellent and successful and to get my master and Ph.D. degrees, and have been putting up with me being away from them for long time to help me achieve that.

This work was supported in part by the Semiconductor Research Corporation, Contract No. 2001-JH-924, and the National Science Foundation, Contract No. CCR-0120369.

Statement of Originality

To the best of my knowledge, the following concepts, ideas, methods, and designs are original and are my work:

1. The synchronous linear inter-pulse-interval (IPI) representation for negative, zero, and positive values.
2. The IPI-to-Voltage conversion method and circuit design based on the representation in 1.
3. The Voltage-to-IPI conversion method and circuit design based on the representation in 1.
4. The addition method and circuit design based on the representation in 1.
5. The subtraction method and circuit design based on the representation in 1.
6. The division method and circuit design based on the representation in 1.
7. The multiplication method and circuit design based on the representation in 1.

Contents

List of Tables

List of Figures

Abstract

Representations and Circuits for Time Based Computation

Khaldoon Mhaidat, B.S., M.S.

Ph.D., OGI School of Science & Engineering
at Oregon Health & Science University

March 2006
Thesis Advisor: Dr. Marwan Jabri

Analog signal representation will remain essential wherever there is a need to interface with the analog world or to satisfy certain area, power consumption, or speed requirements. This includes but is not limited to sensors, instrumentation, and communications. Analog representation is also essential for the integration of analog mixed-signal and RF functions into complex system-on-a-chip (SOC) designs. Today, analog signals are still being represented mainly by current or voltage. Also, data is usually obtained from sensors in voltage or current form. These analog signals are not immune to noise, and therefore they have to be converted into digital for transmission.

In this thesis, we propose a third approach that converts the analog signal into a pulse stream, using time rather than magnitude. This alternative approach uses the inter-pulse time (IPI) to represent the signal values. The thesis will show that our representation approach, unlike the other pulse time representation approaches, is very useful not only in communication but in computation as well. Suitability for both communication and computation is very important because it eliminates the need to convert to/from the analog or digital domains to use their computation techniques if computation is needed. One good example where computation would be needed with communication is the use of averaging at the front end of the receiver to improve the signal-to-noise ratio (SNR). The thesis will also show that our approach is a hybrid approach that takes from digital the immunity to noise, cross-talk, and other problems such as process variations, temperature, and reference voltage, and takes from analog the compactness and low power consumption.

In this thesis, we also present a novel class of methods and circuits for basic conversion and computation based on our novel IPI representation approach above. These methods and circuits include Voltage-to-IPI, IPI-to-Voltage, addition, subtraction, division, and multiplication. We validate these methods and circuits by mathematical derivation, simulation, and chip fabrication and test in CMOS technology. We also compare our IPI implementations versus analog and digital implementations, show their advantages, and discuss how they can be used in applications such as communications, instrumentation, telemetry, signal processing, and ANN's.

1. Introduction

Analog signal representation will remain essential wherever there is a need to interface with the analog world or to satisfy certain design requirements such as power consumption, area, or speed within its noise immunity and accuracy limits [4], [6], and [21]. Example applications include but are not limited to sensors, instrumentation, communications, signal processing, artificial neural networks (ANNs), biomedical actuation, and industrial control. Moore's law and further reduction of the feature size in CMOS IC technology will continue as a technology imperative that drives the cost of electronic products down to mass market level. And now we have system-on-a-chip (SOC) integration which replaces multiple chips of different functions with a single chip [64] and [65]. Such functions may include micro-processing, digital signal processing (DSP), analog mixed-signal and RF functions, and others. A good example for this is biomedical sensors. Analog signal representation is very important for the integration of analog mixed signal and RF functions into such complex SOCs because of problems such as noise, substrate coupling, and cross-talk, which are even more serious for analog circuits [64-68]. These days, analog signals are still being represented in the analog domain mainly by voltage or current signals [21]. Moreover, data is usually obtained from sensors in voltage or current form. These analog signals are not immune to noise and therefore cannot be used for data transfer [4], [6], and [21]. To achieve high immunity to noise, they have to be converted into digital immediately and transmitted digitally. This whole path also has an inverse which would be used, for example, in actuators.

In this thesis, we propose a third alternative approach that converts the analog representation into a pulse stream representation, using time rather than magnitude to represent the signal values. Our alternative approach, as this thesis will show, is very suitable and robust in both communication and computation. The thesis will also show its advantages over traditional pure analog and pure digital representations. Suitability of the representation for both conversion/communication and computation is very important because it eliminates the need to convert to/from other analog or digital domains to use their computation techniques if computation is needed. One good example where computation would be needed with communication is the use of averaging at the front end of the receiver to improve the signal-to-noise ratio (SNR) [25]. Our approach uses the inter-pulse interval (IPI) time to represent the signal values. Figure 1.1 shows one possible way of doing this. The time between each two consecutive pulses encodes a value of the analog signal. The idea of using pulses to encode and process information is not a new one and we are not the first to use it. It is borrowed form neuroscience where information in the brain is encoded using pulses [6], [7], and [21]. The basic framework for research into pulse modulation techniques was laid down around 60 years ago in [71-74] but a significant interest has been experienced with the advent of optical fiber communication systems [101]. Researchers have used pulses in both communication and computation. Section 2.2 of this thesis is dedicated to reviewing their work. We will explain in the thesis how our approach is different from theirs, why it is needed, and how it advances the state of the art.

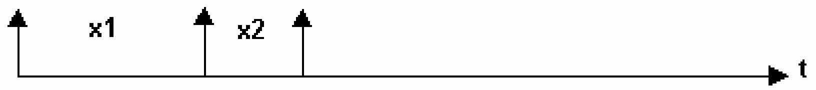


Figure 1.1. Information-carrying inter-pulse intervals (IPIs).

Before we discuss the main motivations for the IPI representation and its advantages over traditional pure analog or pure digital representations, we would like to emphasize that we do not believe it will replace all the existing representations but it will rather be a useful mixed signal technique that is best suited to certain applications or situations. The motivations for using pulses to encode information using time rather than magnitude are discussed in [6], [7], [16], [101], and [102]. When considering communication, the main motivation is that pulses are much more immune to noise than analog signals. Pulses are also much more immune to process variations, temperature, and reference voltage, and to the serious problems that challenge complex mixed-signal SOC integration in deep submicron and nano technologies, such as substrate coupling, cross-talk, transmission line effects, threshold inconsistency, subthreshold currents, hot-electron effects, and doping variability [21] and [64-68]. The IPI representation provides significantly higher immunity to all these problems than traditional voltage and current representations because it encodes the information using the time between the pulses rather than their magnitude [6], [7], [101], and [102]. This is basically converting the analog information to carefully timed signal transitions that are similar to digital schemes. A pulse is detected if it is above a certain voltage threshold, exactly in the same way a binary 0 or 1 value is detected in the on-off digital scheme [105]. Another main motivation is that pulses are much easier to transfer and refresh than analog signals because of their similarity to digital signals, as just described. Pulses can be easily refreshed using digital buffers while analog signals are sensitive to noise and degrade in magnitude especially if they need to travel over a relatively long distance. This makes pulses a much better choice for inter-chip communication [6] and [7], or even for transferring the signal within the same chip if the wire is relatively long or noise or cross-talk, for example, is a concern as in SOC.

When considering issues surrounding local computation, this thesis will show that our IPI computation approach does not have the serious problems that analog computation suffers from such as the body effect and the mobility degradation effect [26], [111], and [1]. When considering issues surrounding global long-wire computation, our IPI approach has all the local computation advantages, as well as the advantages of communication discussed above, since the computation involves communication of the signals over a relatively long distance (long wire).

When compared to representations in the digital domain, the main advantage of the IPI representation, as this thesis will show, is that analog to IPI conversion needs significantly less area and power than analog to digital conversion (ADC). This makes the IPI technology the right choice for applications such as instrumentation, communications, and telemetry if 98% of accuracy, which is equivalent to 5.6-bit digital accuracy, is adequate. The thesis will show that the IPI representation is a hybrid representation that takes good characteristics from both digital and analog representations and blends them together. It takes the high immunity to noise and other problems from the digital and it takes the compactness and low power consumption from the analog [6], [7], [101], and [102].

In chapter 2 of this thesis, we discuss general definitions for the IPI representation and some basic considerations that should be taken into account while creating circuits that take advantage of this representation, and we will briefly review two known IPI conversion schemes, the linear and the logarithmic, and two known IPI signaling schemes: the synchronous and the asynchronous. Analyzing and evaluating these conversion and signaling schemes against each other is carried out in the next chapter when we present our novel representation, to justify the conversion and signaling schemes that we have chosen for our representation. Then, we review the literature and the previous work related to the IPI representation and other pulse representation, modulation, and computation techniques.

In chapter 3, we present a novel IPI representation that is suitable for encoding all values including negative, zero, and positive values, and suitable for all basic conversions and computations as well. This specific IPI representation is synchronous (SIPI) since the time difference is always relative to a reference pulse, and is different than the asynchronous IPI representation (AIPI) as discussed by Ravi and Hammerstrom [22]. We also explain how our new representation is different from the other representations which we review in section 2.2, why it is needed, and how it advances the state of the art. Furthermore, to justify our choice of linear conversion versus nonlinear and synchronous signaling versus asynchronous, we will investigate them and compare them against each other in terms of accuracy, bandwidth, complexity, and suitability for computation. In the last section, we study the conversion and computation timing requirements in general and their effect on the timing of the IPI signal.

In chapter 4, we present novel methods and circuit designs for conversion and computation based on our novel IPI representation in chapter 3. These methods and circuits include IPI-to-Voltage conversion, Voltage-to-IPI conversion, and the basic arithmetic computations: addition, subtraction, division, and multiplication. In light of our explanation in chapter 3 of how our novel representation differs from the other representations in chapter 2 and why it is needed, we explain how our new methods and circuits are different from the other methods and circuits implementing the other representations, why they are needed, and how they also advance the state of the art. Simulation data is shown in this chapter only to show how the circuits operate.

We have fabricated the methods and circuits described in chapter 4 in a chip using the TSMC 0.35 um mixed-signal CMOS process technology, thorough the MOSIS-USA fabrication and packaging service [118] and [119]. In chapter 5, we describe the top level design of the chip including the top level cells. For each of the conversion and computation circuits, we present the simulation and test results at 10 MHz, in terms of accuracy, area, power consumption, and dynamic range. We also present the simulation results for conversion, addition, and subtraction at 100 MHz, and for division and multiplication at 50 MHz.

In chapter 6, we describe a system-level design which performs the functions of a neuron with multiple synapses in ANNs. The design incorporates the basic arithmetic building blocks that we present in chapter 4 and operates them together as a system. We also provide the system-level simulation results at 50 MHz. We also discuss how the IPI technology can be used in applications such as instrumentation, communications, and telemetry, and signal processing. To demonstrate the advantages of using the IPI technology in these applications, we compare our IPI based conversion and computation implementations with other analog and digital implementations.

In chapter 7, we summarize the importance of this work, and discuss future areas of research.

2. Pulse Representations

In this chapter, we will present general definitions of the IPI representation and some basic considerations that should be taken into account while developing an IPI representation, and we will briefly review two known analog-to-time conversion schemes, the linear and the logarithmic, and two known signaling schemes, the synchronous and the asynchronous. Then, we will review the literature and the previous work related to the IPI representation and other pulse representation, modulation, and computation techniques.

2.1. The IPI Representation

2.1.1. General Definitions for the IPI Representation

In this section, we define the IPI representation in general terms. Here are definitions of the terminology and the notations that we will use:

- t_i = pulse start time
- x_i = analog value represented by the IPI time difference between the pulse starting at t_i and the reference pulse
- t_w = pulse width time
- x_{min} = minimum analog value
- x_{max} = maximum analog value

- $f(x)$ = analog-to-IPI conversion function
- $f^{-1}(x)$ = IPI-to-analog conversion function
- $t_{min} = f(x_{min})$
- $t_{max} = f(x_{max})$
- δ = time error in conversion. This defines the resolution or accuracy of time measurement. We will see in chapter 3 that this will help us evaluate the effect of the conversion function on the computation errors.

Factors like noise, accuracy, speed, and practical design limitations will affect how short the shortest IPI can be.

2.1.2. Basic Considerations in Analog-to-IPI Conversion

It is very important to carefully examine the analog-to-IPI conversion function before developing the IPI representation itself or designing any IPI hardware, because it directly impacts the conversion and computation methods and the complexity, accuracy, speed, power consumption, and area of their circuit implementations. Any analog-to-IPI conversion function should meet the following criteria:

1. Can be efficiently implemented in available IC technologies, represented by CMOS.
2. Is reversible, that is, the inverse function can also be implemented in available IC technologies, represented by CMOS.
3. Is continuous, to achieve better accuracy by avoiding the quantization error. If we consider two versions of the same analog-to-IPI mapping function, one continuous and another discrete (quantized), then the continuous version will have better resolution than the discrete one and therefore better accuracy. This

is not an absolute requirement and the designer may choose a discrete function for a number of reasons.

4. Can be used to represent all values including negative, zero, and positive. If it does not meet this criterion, it will be considered as incomplete.

2.1.3. Conversion Schemes: Linear and Logarithmic

In the linear representation, the time t is linearly proportional to the encoded analog value x [69] and [70]. The general form of the linear conversion function is

$$t = f(x) = ax + b \qquad (2.1)$$

where a and b are real constant numbers. In the nonlinear representations such as the hyper-tangent or the logarithmic [7-10], the relationship between the time and the analog value is not linear. In the logarithmic representation, for example, the general form of the conversion function is

$$t = f(x) = a \log_b(x) + c \qquad (2.2)$$

where a, b, and c are real constant numbers. A more general form of the logarithmic function that can be used to deal with the cases when x is zero or negative is

$$t = f(x) = a \log_b(x + d) + c \qquad (2.3)$$

where d is a real positive constant that can be added to x so that the input to the log function is always positive. This constant d needs to be accounted for when converting back from the time domain to the analog domain. Analyzing and evaluating the linear and logarithmic schemes relative to each other will be carried out in the next chapter as part of explaining why the novel IPI representation scheme, which is also presented in the next chapter, is suitable for computation.

2.1.4. Signaling Schemes: Asynchronous and Synchronous

Figure 2.1 shows two known signaling schemes: *asynchronous*, as in [83] and [94-97], and *synchronous* as in [69], [70], [75-81] and [98]. In the asynchronous scheme, information is encoded in each IPI. The pulses are asynchronous because the IPI time is completely dependent on the encoded value and is measured between a pulse and the pulse immediately preceding it. The synchronous scheme imposes more constraints on the timing of the pulses by using a framing clock pulse as a reference. One full time frame T_f is dedicated to each value to be encoded. The time between the two framing pulses define the time frame T_f. In its simplest form, the IPI time is the time between the first framing pulse and the data pulse that appears between two framing pulses.

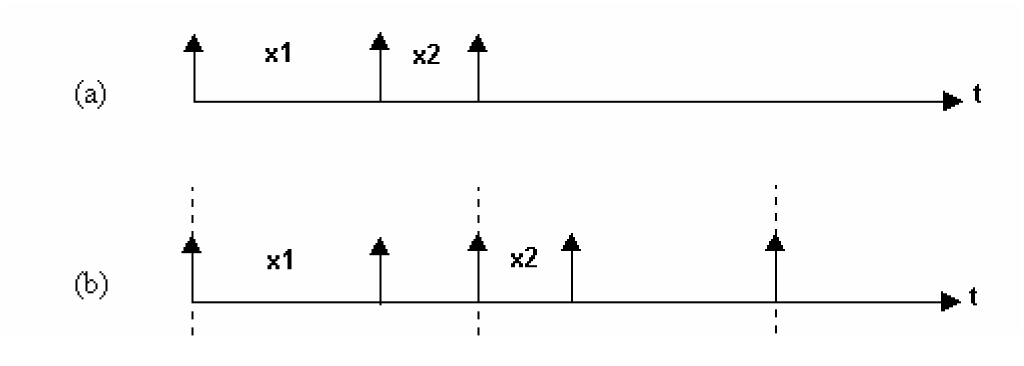


Figure 2.1. IPI signaling schemes: (a) Asynchronous (b) Synchronous.

2.2. Literature Review

Although we have very few ideas that are closely related to our work, in this section, we will review the literature and the previous work on other pulse representation, modulation, and computation techniques. In fact, one of the main challenges with the research described here is the shortage of previous work which could directly help us, considering our objective of developing an IPI representation suitable for all values (negative, zero, and positive) and all basic conversion and computation operations at the same time. This has made our research problem more interesting and challenging. To the best of our knowledge, this thesis is the first to develop a novel IPI representation that is suitable for encoding all values including negative, zero, and positive, and then to use it to develop a class of novel time-based methods and circuits for all basic conversion and computation.

2.2.1. Pulse Modulation Techniques

The idea of using pulses to encode and process information is not a new one. It is borrowed form neuroscience where information in the brain is encoded using pulses [6], [7], and [21]. The basic framework for research into pulse modulation techniques was developed around 60 years ago in [71-74] but a significant interest has been experienced with the advent of optical fiber communication systems [101]. Reviews and discussions of various pulse modulation techniques and how they can be used to encode information can be found in [6], [7], and [101]. Figure 2.2 shows four main techniques. In pulse amplitude modulation (PAM), the input signal modulates only the amplitude of the pulse. In pulse width modulation (PWM), the input signal modulates only the width of the pulse. Some literature also calls this pulse length modulation or pulse duration modulation. In pulse delay modulation (PDM), the input signal modulates only the delay of the pulse with respect to another pulse. In some literature, the synchronous version of this technique is called pulse position (PPM) modulation and the asynchronous version of it is called pulse interval modulation (PIM). In pulse frequency modulation (PFM), the input signal modulates only the frequency of the pulse. Some literature also calls this pulse rate modulation. To be consistent, we will refer to PDM as inter-pulse interval modulation (IPI-M) or simply IPI throughout the thesis. Another technique which is not shown in the figure is called pulse-code modulation (PCM). PCM is basically a digital representation in which a single analog sample is encoded using multiple binary-weighted pulses (bits). PWM, IPI, PFM, and PCM are significantly more immune to noise than PAM since it uses the amplitude of the pulse to encode the information while they use only time-dependent features of the pulses [6], [7], [101], and [102].

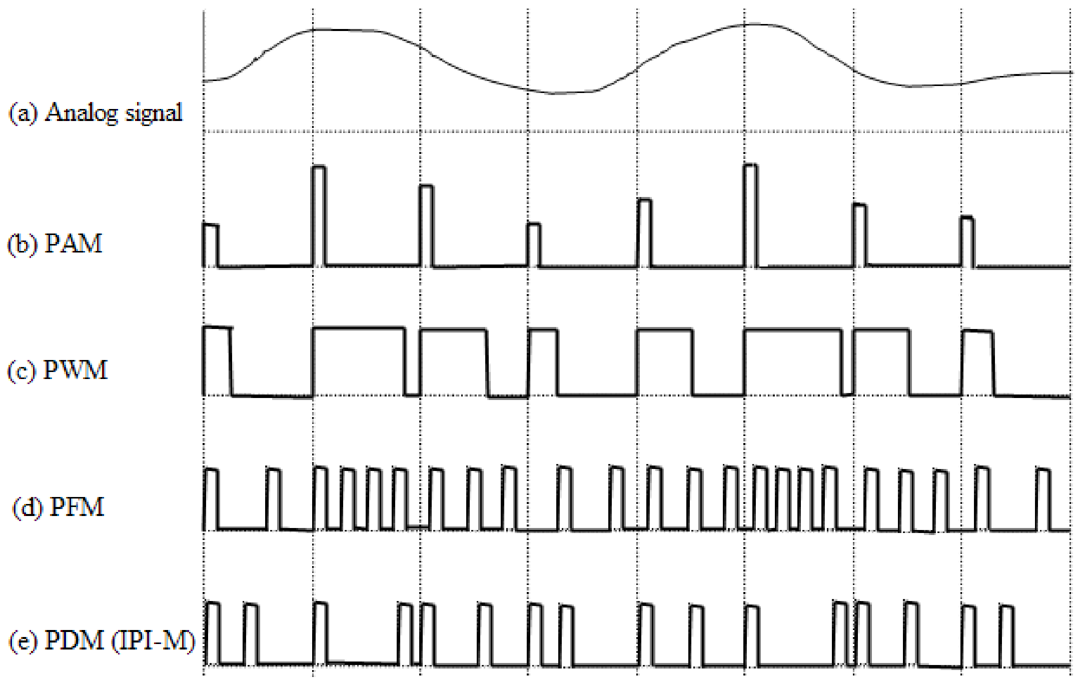


Figure 2.2. Pulse modulation techniques: (a) the analog signal, (b) pulse amplitude modulation (PAM), (c) pulse width modulation (PWM), (d) pulse frequency modulation (PFM), and (e) pulse delay (or inter-pulse interval) modulation (PDM or IPI-M).

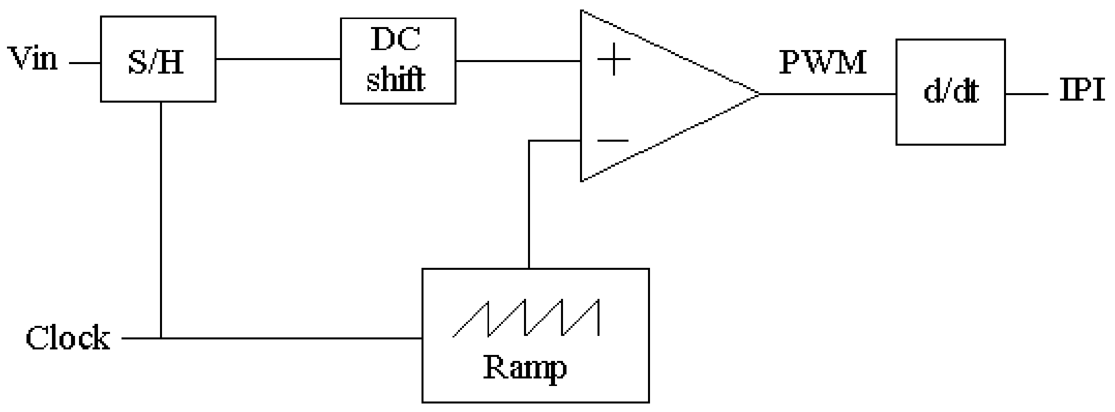


Figure 2.3. Synchronous IPI and PWM modulation. In asynchronous IPI and PWM, no clock is used and S/H and the ramp are restarted as soon as the comparator detects the equivalence between the input sample and the ramp.

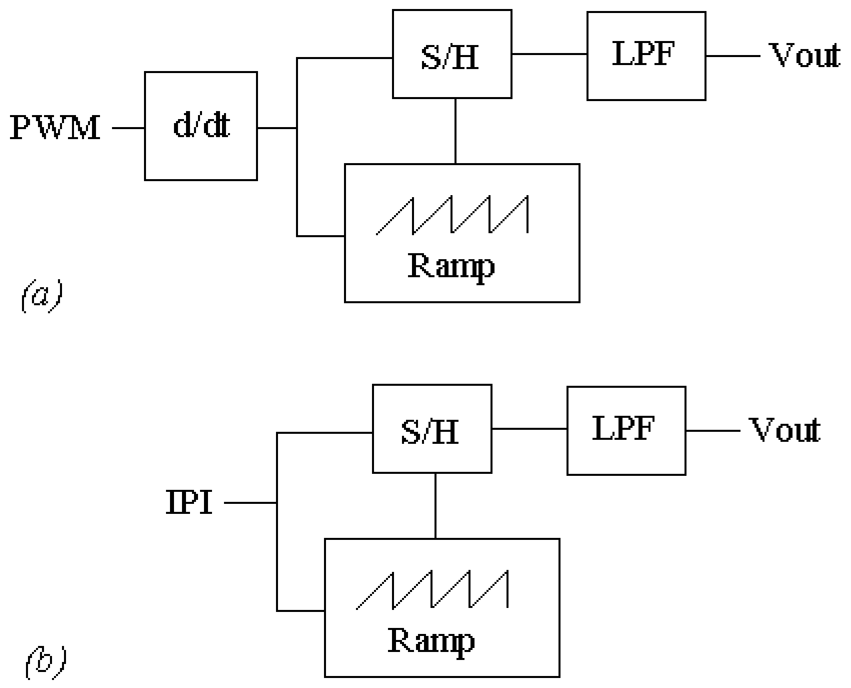


Figure 2.4. IPI and PWM demodulation: (a) PWM and (b) IPI.

Riter et al. in [69] and [70] give a block level description of a synchronous IPI modulation technique which they use for transmitting information via underwater acoustic channel. It is similar to the block diagram in Figure 2.3. Pulse modulation techniques have been adapted for use in optical fiber transmission of analog and video signals: Synchronous PWM [75-81] and [98], PFM [82], [84-93], and [103], and asynchronous IPI and PWM [83] and [94-97]. IPI and PWM are very similar and can be easily obtained from each other. The IPI signal is obtained from the PWM signal by differentiating the PWM signal to generate narrow pulses at the edge transitions. The PWM signal is obtained from the IPI signal by simply using a bistable circuit like a latch or a flip-flop. All the IPI and PWM applications in [69], [70], [75-81], [98], [83], and [94-97] use a modulator and a demodulator similar to the ones depicted in Figures 2.3 and 2.4, respectively. The modulator samples and holds (S/H) the analog signal and compares the sample with a sawtooth ramp signal. If equivalence is detected then the comparator output goes from high to low. The PWM signal is taken from the output of the comparator. The IPI signal is obtained by differentiating the PWM signal. In synchronous IPI and PWM, a periodic clock is used to control the timing of the S/H operation and the ramp generation. In asynchronous IPI and PWM, however, no clock is used and the S/H operation and the ramp generation are restarted as soon as the comparator detects the equivalence between the input sample and the ramp. The input signal has to be DC-shifted to accommodate the most negative value. Otherwise, it will be limited by the ramp minimum voltage and the comparator input range.

Demodulation of PWM is performed by converting the PWM signal to IPI first and then performing IPI demodulation as shown in Figure 2.4. For IPI demodulation, the first input pulse is used to initiate the generation of a sawtooth ramp signal and the second input pulse is used to sample and hold the ramp signal. This produces the voltage sample equivalent to the IPI. A low pass filter (LPF) can then be used to recover the baseband signal component from the frequency spectrum [101]. The ramp signal used in all the IPI and PWM applications above is linear, which leads to a linear relationship between the voltage signal and the IPI and the pulse width. It is possible to use a nonlinear ramp instead if the application requires that. For example, Murray et al. in [7-10] use a sigmoidal ramp in order to model the nonlinear and squashing properties of the neuron transformation function in ANNs, as we will see in the next subsection. IPI consumes less power than PWM since it encodes the information in the time between fixed-width narrow pulses while PWM encodes it in the variable width of the pulses [101].

For PFM modulation, [82], [84-93], and [103] use a voltage-controlled oscillator (VCO) followed by a monostable circuit to generate fixed-width narrow pulses. For demodulation, they use a monostable circuit to reconstruct fixed-width narrow pulses from the input stream, followed by a LPF to recover the base-band signal component from the frequency spectrum [101].

2.2.2. Pulse Computation Techniques

All the pulse based computation circuits and techniques we have found reported in the literature deal mainly with two operations: addition and multiplication. The reason is that these two operations, as we will see in chapter 6, are heavily used in many applications like ANNs and signal processing. We will first discuss the addition techniques then the multiplication ones.

A logical OR gate can be used to add PFM pulse streams together [6]. The output will also be a PFM pulse stream. This is a very cheap solution but its accuracy decreases as the pulse overlap increases. Therefore, it is suitable for a small number of inputs with a very small Mark-Space ratio (High time to Low time ratio) to reduce the probability of pulse overlap. Murray and Smith in [11-13] succeeded to achieve 97% accuracy because the number of inputs to the OR gates (in a small ANN of only 10 neurons) was small and the maximum Mark-Space ratio was 0.01.

The pulse overlap problem in the technique above can be avoided by treating the pulses as current pulses instead of voltage pulses [6]. Current pulses add together if they overlap producing more current unlike voltage pulses going into an OR gate which will count as one pulse if they overlap. The total current can then be integrated as charge on a capacitor according the following equation

$$V_C = \frac{Q}{C} = \frac{\int Idt}{C} \qquad (2.4)$$

The output of this current integration is a voltage. To convert it to a PFM pulse stream for example, a voltage-controlled oscillator (VCO) can be used, followed by a monostable circuit to generate fixed-width narrow pulses.

Murray et al. in [7-10] present pulse stream computation techniques and apply them to ANNs. An introduction to ANNs can be found in section 6.1. The operation of a neuron with its synapses can be modeled by the following equation

$$y = f\left(\sum_{i=1}^{N-1} w_i \cdot x_i\right) \qquad (2.5)$$

where x_i is an input, w_i is the synaptic weight for that input, and f is the neuron transformation function. A common function is the sigmoid function

$$f(sum) = \frac{1}{1+e^{-sum}} \qquad (2.6)$$

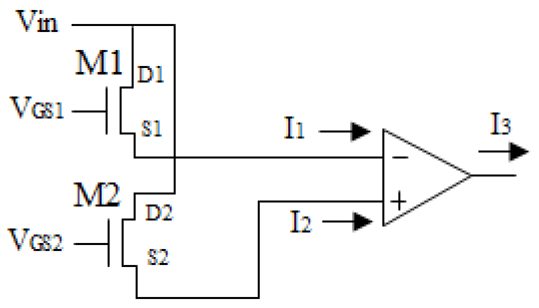


Figure 2.5. The transconductance multiplier.

In (2.5), the synapse performs the multiplication and the neuron performs the summation and the transformation. The synapse design that was used in [7-10] uses the analog transconductance multiplier described in [14] and [15]. Figure 2.5 shows the transconductance multiplier. As a first order approximation, the drain current of the MOSFET transistor when operating in the triode region can be calculated as

$$I = \beta\left[\left(V_{GS} - V_t\right)V_{DS} - \frac{V_{DS}^2}{2}\right] \qquad (2.7)$$

where V_{GS} is the gate to source voltage, V_{DS} is the drain to source voltage, V_t is the threshold voltage, and β is the gain. If the two transistors M1 and M2 are identical and V_{DS1} is equal to V_{DS2} then

$$I_3 = I_1 - I_2 = \beta\left(V_{GS1} - V_{GS2}\right)V_{in} \qquad (2.8)$$

Now we can see that the resultant current is a scaled value of the multiplication of two voltage quantities. The undesired nonlinear components in the individual drain currents have been cancelled by subtracting one from the other. Nonlinearity cancellation by addition or subtraction is a common technique in almost all analog multipliers.

Figure 2.6 shows the pulse stream transconductance multiplier used in [7-10]. The weight w_i is stored as a voltage on a capacitor and refreshed periodically from an external random access memory (RAM) using a digital to analog converter (DAC). The synapse uses the transconductance multiplier to convert the weight voltage into current. The input x_i is a pulse stream. It can be a PFM or a PWM stream. It controls the transistor M3 by turning it on when the pulse is high and off when the pulse is low. The amplitude of the current pulse through M3 when it is on is proportional to w_i and its frequency and duration is the same as the input pulse stream x_i. Therefore, the synaptic weight w_i performs pulse amplitude modulation (PAM) while the input pulse stream x_i performs pulse frequency modulation (PFM) and pulse width modulation

(PWM). The operational amplifier (OP-AMP) in negative feedback with the transistor buffers M4 and M5 provide stronger current drive to support current demand from many other synapses that connect to the OP-AMP. It converts the current pulses into voltage pulses *Vouti*. It is then followed by a voltage integrator to aggregate these voltage pulses. To convert the voltage integrator output to a PFM signal, a VCO is used. Higher voltage level generates more oscillations (pulses). To convert the integrator output to a PWM signal, a comparator with a sigmoidal ramp connected to its negative input is used. Higher voltage level generates a wider pulse since the sigmoidal ramp will be below it for longer time. The sigmoidal ramp is provided to the chip externally and it provides the desired nonlinear and squashing behavior of the neuron transformation function.

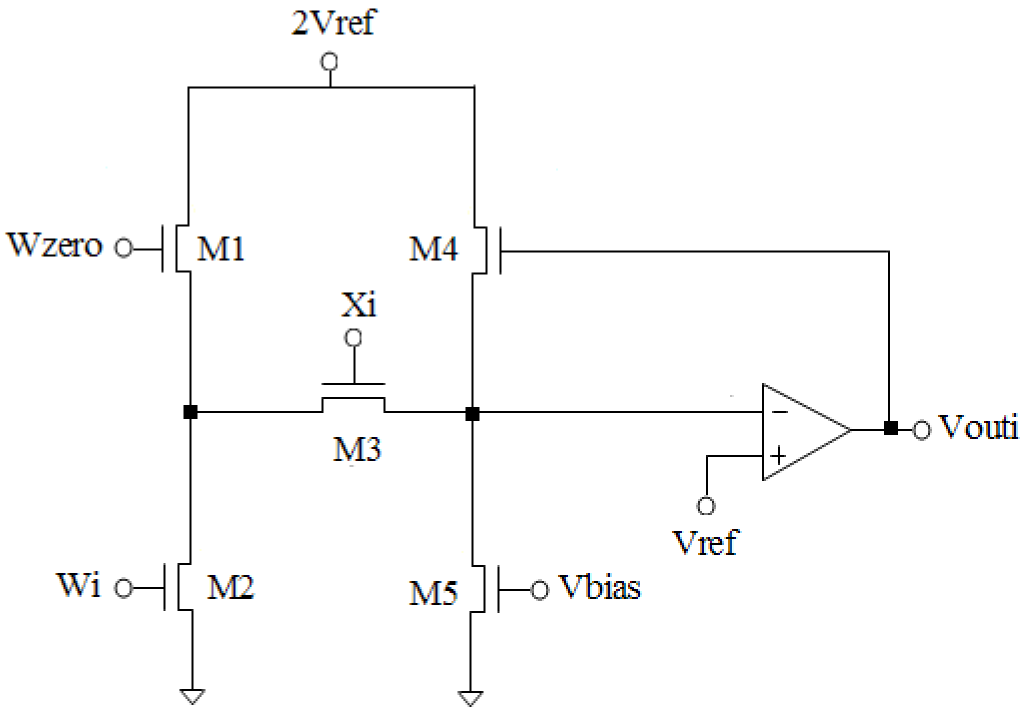


Figure 2.6. The pulse stream transconductance multiplier.

Del Corso et al. in [16-18] present a mixed digital/analog technique which incorporates both PFM and PWM techniques and applies them to ANNs. The input pulse stream x_i is coded in its frequency and used for PFM while the weight w_i is coded as a binary number in a shift register and used for PWM. When a pulse comes from the input x_i, it triggers a full rotation of the shift register. The shift register clock has a varying binary-weighted cycle time ($T, 2T, 4T, 8T, \ldots, 2^{N-1}T$) where N is the bit resolution of w_i. This will let a bit stay at the output of the shift register for time proportional to its binary weight (its position in the shift register). If the output bit is '1' then it enables a current switch and generates a current pulse of width equal to its time. The total width of the current pulses is proportional to w_i. The current pulses charge a capacitor and its voltage is therefore proportional to w_i. w_i is a 2's compliment number and the sign bit is used to direct the current in or out of the capacitor (in the negative or positive direction) based on its value, 1 or 0, respectively. Since this process is repeated at the input pulse frequency (rate) then the capacitor voltage is also proportional the input pulse frequency. The output voltage drives a VCO to convert it to a PFM signal.

Mead in [5] presents a self-resetting integrate-and-fire neuron design. It is similar to the integrate-and-fire circuit shown in Figure 2.7. From its name, the integrate-and-fire technique integrates the input current I as a voltage on the capacitor C. When the voltage reaches above the comparator threshold, it fires a pulse. The pulse then turns the transistor M3 on. This causes the reset signal to go high which disconnects the input and discharges the capacitor to be ready for a new cycle. More current means the capacitor will charge and reach the threshold faster. This means that the time between the generated pulses will be shorter. This integrate-and-fire technique is an IPI modulation technique since the input current modulates the inter-pulse time interval (IPI). IPI is inversely proportional to the input current. It can also be thought of as a PFM technique because shorter IPIs mean a higher pulse frequency.

Winner-take-all (WTA) networks like the ones in [47] and [48] rely on the integrate-and-fire technique. The current source and inverter in the dashed box in

Figure 2.7 are common to all neurons or cells in the network. Once the first neuron or cell reaches the threshold, the reset signal resets all of them. This means that the cell with the maximum input will win and shut-off all the other cells. This is useful in sensory array applications when the interest is focused on the sensor with the strongest input. Because of its parallel processing capability, it is much faster to solve this problem using a dedicated on-chip WTA network than to solve it using some maximum search algorithm.

One useful asynchronous time-domain pulse-stream communication protocol is called the address-event representation (AER). It is described in [49] and [50]. It allows multiple pulse senders, such as winner-take-all networks or sensor arrays, to share the same line. To signal an event (pulse), the sender requests access to the transmission line and puts its binary address on it when access is granted. On the receiver side, the binary address is decoded to determine the source (sender) and the pulse is reconstructed. The inter-pulse time for a sender is measured between two consecutive assertions of its address.

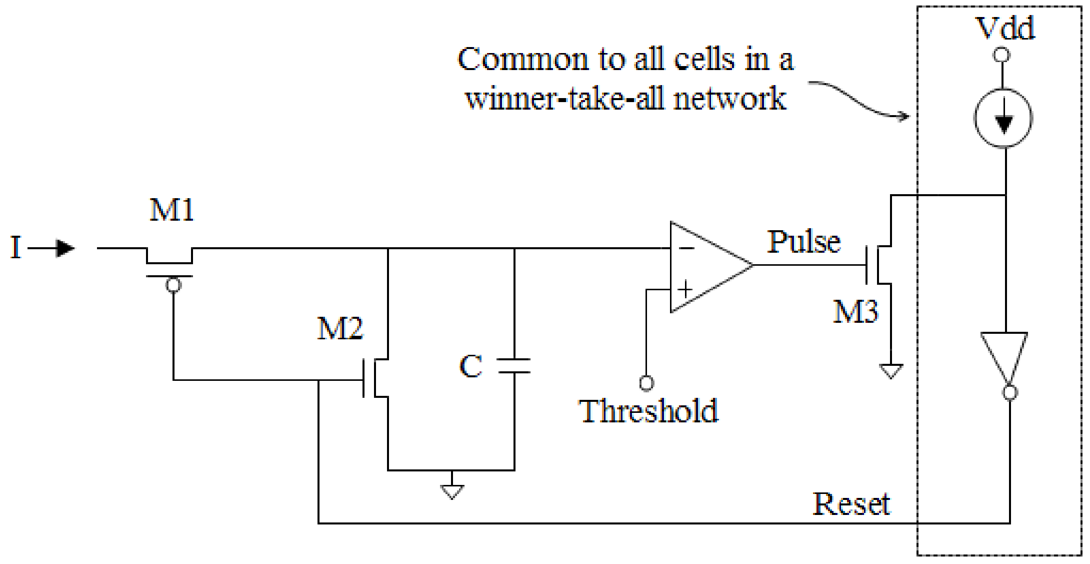


Figure 2.7. Self-resetting integrate-and-fire circuit. The current source and the inverter in the dashed box are common to all cells in winner-take-all (WTA) networks.

Culurciello et al. in [51] present a fully-arbitrated digital imager based on the integrate-and-fire and the AER techniques. The photosensor generates a current that is linearly proportional to the light intensity. Therefore, the inter-pulse time is inversely proportional to the light intensity. The imager has higher bandwidth than traditional imagers and a superior signal to noise ratio. The higher bandwidth is because it favors (gives more bandwidth to) pixels with higher intensity, unlike traditional imagers which scan the pixel array serially and give equal bandwidth to each pixel regardless of its intensity. It also consumes less power because there is no serial scanner that is active all the time and because the pixel's power consumption depends on its activity (intensity). Bright pixels consume more power than dark pixels. Abrahamsen et al. in [48] present a motion detection imager based on the AER and the WTA techniques above. Similar to the imager in [51], the inter-pulse time is inversely proportional to the light intensity.

3. A Novel Approach to IPI Representation

In this chapter, we will present a novel IPI representation that is suitable for encoding all values including negative, zero, and positive values, and also suitable for all basic conversions and computations. We will also explain how our new representation is different from the other representations we reviewed in section 2.2, why it is needed, and how it advances the state of the art. Furthermore, to justify our choice of linear versus nonlinear conversion, and synchronous versus asynchronous signaling, we will compare them against each other in terms of accuracy, bandwidth, complexity, and suitability for computation. In the last section, we will study the conversion and computation timing requirements in general and their effect on the timing of the IPI signal.

3.1. A Novel IPI Representation

The IPI representation we are presenting here is linear and synchronous. It is linear because the relationship between the value and the IPI time representing it is linear. It is synchronous because the pulses are referenced with respect to a periodic framing or reference clock signal which also controls the conversion and computation. Figure 3.1 shows the representation and how it encodes negative, zero, and positive values. T_f is the frame time. The time between the first framing pulse and the pulse in between is defined as t_+. The time between the pulse in between and the second framing pulse is defined as t_-. If t_+ is greater than t_- then the value is positive. If t_+ is equal to t_- then the value is zero. And if t_+ is less than t_- then the value is negative. The relationship between T_f, t_+, and t_- is governed by the following equation

$$t_+ = T_f - t_- \tag{3.1}$$

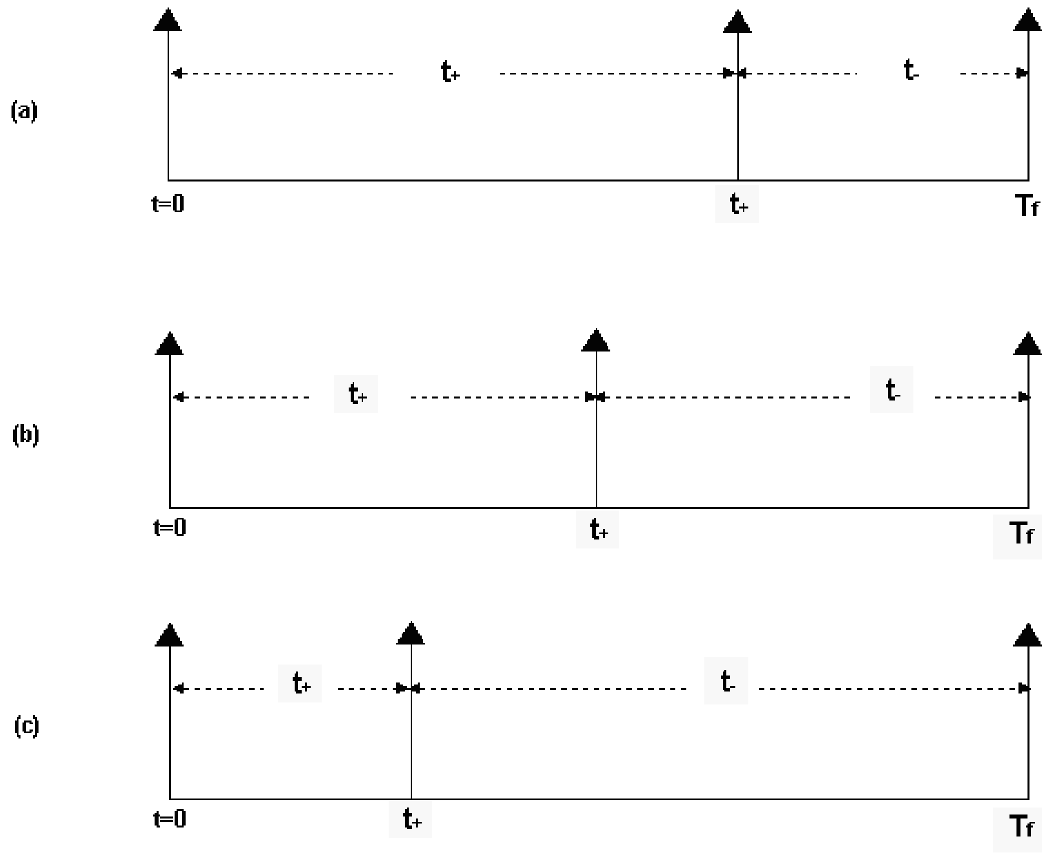


Figure 3.1. Novel synchronous linear IPI representation for negative, zero, and positive values: (a) $t_+ > t_-$ for positive, (b) $t_+ = t_- = T_f/2$ for zero, and (c) $t_+ < t_-$ for negative.

When compared to other pulse representations, our representation is novel in the way it encodes negative, zero, and positive values. The representations [69], [70], [75-81] and [98], are all linear and synchronous, but these representations use only one part of the frame (the time between the framing pulse and the pulse in between), let us call it t, while our representation uses both parts t_+ and t_-. I.e., our representation introduces the concept of negative time while the other representations do not have such concept. This makes our representation more suitable for computation of negative, zero, and positive values because it allows for "time-based" computation in the time domain with no need to convert to the analog or digital domains for computation, as we will explain.

This thesis shows, in subsection 6.2.1, how time-based computation using our representation, not just communication, is more robust than analog computation. It is true that the other pulse time representations can still represent or convert negative values or any range of the signal by using a sawtooth ramp signal and/or DC offset of the input signal relative to the comparator, as we have described in section 2.2. But, direct time-based computation using these representations is more difficult. Let us use addition as an example. Consider the following simple linear function which is similar to the voltage-time relationship used in [69], [70], [75-81] and [98]. For simplicity, we will assume the slope is 1 since it has no impact on the discussion.

$$x = t - B \tag{3.2}$$

where x is the analog value, t is the time representing it, and B is an offset that allows the representation of negative values. This offset can be achieved by letting the sawtooth ramp start from negative and/or by adding a DC offset to the input signal. On the other hand, the following function is similar to the representation that we are proposing. Again for simplicity, we will assume the slope is 1.

$$x = t_+ - t_- \tag{3.3}$$

Now, let us consider the task of adding three analog values in the time domain. The values are x_1, x_2, and x_3. Using (3.2), we can find

$$t_1 + t_2 + t_3 = x_1 + x_2 + x_3 + 3B \qquad (3.4)$$

but using (3.3), we can find

$$(t_{1+} - t_{1-}) + (t_{2+} - t_{2-}) + (t_{3+} - t_{3-}) + = x_1 + x_2 + x_3 \qquad (3.5)$$

Equation (3.4) which uses the other representations approach, shows that if we add the times representing the three analog values then the result will be a time that represents their sum plus an offset that is equal to the sum of their individual offsets. In this case the offset is *3B* and *N* inputs will have an offset of *NB*. This offset then needs to be subtracted from the result. Time-based subtraction will have the same problem but the offset will be negative not positive. Each addition/subtraction circuit will need to subtract/add a different offset value based on the number of inputs and their signs, so that the result can be delivered to the next time-based computation block offset-free. These offsets will limit the dynamic range of operation as they grow in either direction. They will also impact the accuracy and increase the cost of computation and calibration circuitry. The offset problem is even more serious when we consider time-based multiplication and division. In multiplication, for example, the offsets will get mixed (multiplied) with the inputs and removing them from the final result will be difficult.

Equation (3.5), which uses our representation approach, shows that the time resulting from adding and subtracting the t_+ and t_- times of the individual inputs, does not have the DC offset problem described above and therefore no DC shift or adjustment of the result is needed. The same thing applies to the other computations, multiplication and division. Using this approach, all computation blocks can use the same representation, and will therefore be able to directly receive and generate results in the same form. It is true that from (3.1), equation (3.3), which uses our approach, can be rewritten as

$$x = 2t_+ - T_f \qquad (3.6)$$

which is similar to (3.2), and which uses the other approach, if we ignore the slope of 2 and think of T_f as the offset. Equations (3.3) and (3.6) are equivalent from a conversion and representation perspective. However, the difference is that (3.6) uses the t_+ and T_f information to represent the signal value instead of the t_+ and t_- information used by (3.3). If we choose to use the t_+ and T_f information instead of the t_+ and t_- information then we will run into the same DC offset problem that exists in the other approach, as can be seen from the similarity between (3.2) and (3.6).

Based on our t_+-t_- representation approach, we were able to develop methods and circuits for all basic time-based conversions and computations, including Voltage-to-IPI, IPI-to-Voltage, addition, subtraction, division, and multiplication. Suitability of the representation for both conversion/communication and computation is very important because it eliminates the need to convert to/from other analog or digital domains if computation is needed. One good example where such computation would be needed is at the front end of a receiver to improve the signal-to-noise ratio (SNR) by averaging the over-sampled signal [25]. If the pulse time representation in [69], [70], [75-81] and [98], is used for communication, then to perform computation, the pulse stream signal has to be converted to either analog or digital in order to use the analog or digital computation techniques. This thesis will show that computation based on our approach here does not have the problems that analog computation suffers from such as the body effect and the mobility degradation effect, just considering local short-wire computation issues. Converting from time (IPI or PWM) to digital will require a fast counter and clock. The thesis will show that our approach has accuracy better than 98%, which is comparable to 6-bit digital accuracy, using a 10 ns time frame. Assuming that the other representations in [69], [70], [75-81] and [98] have the same accuracy as ours for the same time frame, to convert from them to the digital domain while maintaining the same level of accuracy and speed, a 6-bit counter with a clock cycle as short as 10/64 = 0.15625 ns is necessary. The clock cycle time required is almost as short as the delay of a single fast inverter cell in the TSMC

0.35um process [117]. So, such a fast counter is not actually possible and therefore we need to make the time frame much longer to accommodate the time-to-digital conversion. This will, of course, slow the speed down significantly, and it will also consume much more power.

It is possible to convert from the time domain back to the analog domain, and then from the analog domain to the digital domain using a fast ADC, to avoid the very fast counter problem. However, this thesis shows that the ADC operation is much more expensive in terms of area and power consumption than Voltage-to-IPI conversion. After all these conversion/computation scenarios, we can see that performing both conversion/communication and computation in the same time domain using the same representation, is the best option if 98% accuracy is enough and area and power consumption are important.

3.2. Computation Error Analysis

The objective of this section is to examine in more depth the linear and logarithmic representations that were introduced in the previous chapter and their effect on error accumulation in various computations. The error analysis results from this section will be used to compare the linear and logarithmic representation schemes in the next section. The errors we will calculate are:

- e_c , conversion error
- e_a , N-input addition error
- e_s , N-input subtraction error
- e_m , N-input multiplication error
- e_d , N-input (one dividend and N divisors) division error

3.2.1. Linear Representation

We will use the following simple linear conversion function to evaluate the effect of the linear function on the error when performing various arithmetic operations in the time domain

$$t = x \tag{3.7}$$

The maximum conversion error is determined by the time resolution which is δ. Therefore,

$$e_{c,max} = (t + \delta) - t = \delta \tag{3.8}$$

Since each one of the N inputs to the addition or subtraction will contribute a maximum error of δ to the total error, the maximum addition and subtraction errors are

$$e_{a,max} = e_{s,max} = N\delta \tag{3.9}$$

To evaluate the multiplication error of N inputs, we will first use a binomial expansion to evaluate the following formula

$$(t+\delta)^N = \sum_{i=0}^{N} \binom{N}{i} \delta^i t^{N-i} = t^N + \sum_{i=1}^{N} \binom{N}{i} \delta^i t^{N-i} \tag{3.10}$$

where

$$\binom{N}{i} = \frac{N!}{(N-i)!\,i!} \tag{3.11}$$

Using (3.10), we can find that the multiplication error is

$$e_m = (t+\delta)^N - t^N = \sum_{i=1}^{N} \binom{N}{i} \delta^i t^{N-i} \qquad (3.12)$$

and therefore the maximum multiplication error is

$$e_{m,\max} = \sum_{i=1}^{N} \binom{N}{i} \delta^i t_{\max}^{N-1} \qquad (3.13)$$

The division error is

$$e_d = (1/t^N) - (1/(t+\delta)^N) = (1/x^N) - (1/(x+\delta)^N) \qquad (3.14)$$

The maximum division error will depend on the smallest magnitude allowed for the divisors.

3.2.2. Logarithmic Representation

We will use the following simple logarithmic conversion function to calculate the error when performing various arithmetic operations

$$t = log_B(x) \qquad (3.15)$$

which means

$$x = B^t \qquad (3.16)$$

The conversion error is

$$e_c = B^{(t+\delta)} - B^t = B^t\,(B^{\delta} - 1) = x\,(B^{\delta} - 1) \qquad (3.17)$$

and the maximum conversion error is

$$e_{c,max} = x_{max} (B^{\delta} - 1) \qquad (3.18)$$

In the same way, we can find that the maximum error of addition and subtraction is

$$e_{a,max} = e_{s,max} = Nx_{max} (B^{\delta} - 1) \qquad (3.19)$$

The multiplication error is

$$e_m = (B^{(t+\delta)})^N - (B^t)^N = B^{Nt} (B^{N\delta} - 1) = x^N (B^{N\delta} - 1) \qquad (3.20)$$

and the maximum multiplication error is

$$e_m = (x_{max})^N (B^{N\delta} - 1) \qquad (3.21)$$

The division error is

$$e_d = 1/(B^{(t+\delta)})^N - 1/(B^t)^N = B^{-Nt} (B^{-N\delta} - 1) = x^{-N} (B^{-N\delta} - 1) \qquad (3.22)$$

The maximum division error will depend on the smallest magnitude allowed for the divisors.

3.3. Conversion Schemes: Linear Versus Logarithmic

The IPI representation we presented in section 3.1 is linear. There are a number of reasons to use a linear scheme over a logarithmic scheme. One of the most important is accuracy. One advantage of the logarithmic scheme is that it provides more bandwidth and therefore more speed than the linear representation. It also provides greater dynamic range, since the logarithmic function compresses the IPI time needed to encode the analog signal, that is, it represents the same signal with shorter IPI intervals. As the base of the logarithm b increases, the IPI time decreases which increases bandwidth. Unfortunately the price for this is mainly in reduced accuracy. From the error analysis and equations in the previous section, we can see that the amount of error in conversion and computation increases as the base of the logarithm *B* increases. The linear scheme does not have this problem and therefore we have chosen it to achieve better accuracy.

The second reason why we have chosen a linear scheme over a logarithmic is that addition and subtraction are more difficult to perform in the logarithmic domain than in the linear domain. We were able to develop novel methods and circuits to perform addition and subtraction using the linear scheme, as we will see in the next chapter. One potential advantage of using the logarithmic scheme however is that multiplication and division can be performed as addition and subtraction respectively, but this potential advantage is compromised by the lack of simple methods and circuits to do addition and subtraction. Furthermore, we were able to find novel methods and circuits to perform division and multiplication using the linear scheme, as we will see in the next chapter.

The third reason is that it is difficult to deal with negative, zero, and positive values less than 1 in the logarithmic domain because negative and zero are out of the input range of the logarithmic function, and positive values less than 1 requires a negative IPI representation. To overcome this problem, a DC offset may be added in the beginning when converting from analog to IPI and then the same offset needs to be subtracted if IPI to analog conversion is needed. The impact of the DC offset on the computation result will depend on the type of operation being performed. For example, in the case of addition, the offset in the result will be the sum of the offsets in the individual inputs. When multiplying two inputs, since the offset of one input will get multiplied by the other input, isolating the offset will be difficult. Consequently, using a DC offset to solve the initial problem of not being able to represent negative, zero, and positive values less than 1 in the logarithmic domain can lead to significantly greater complexity especially in the case of multiplication and division.

3.4. Signaling Schemes: Synchronous Versus Asynchronous

The IPI signal representation presented in section 3.1 is synchronous. We have chosen the synchronous version since it meets our needs for both communication and computation. In this section we discuss the reasons for this decision.

In the asynchronous scheme, information is encoded in each IPI and the pulses are asynchronous with each other. Time-based computation using this scheme is very difficult especially on negative values because of the DC offset problem described in section 3.1. It is possible to convert the asynchronous IPI streams to analog voltages and then use analog computation techniques, but this thesis will show that time-based computation using our approach does not have the problems that analog computation suffers from such as the body effect and the mobility degradation effect. Our synchronous representation is more suitable for performing time-based computation on all values including negative, zero, and positive, because it uses both parts of the frame to represent the value, and this approach is not possible with the asynchronous scheme.

The synchronous scheme imposes more constraints on the arrival times of the pulses by using a framing clock pulse as a reference. One full time frame T_f is dedicated to each value to be encoded, which creates a natural trade-off between accuracy and responsiveness. Shorter intervals are more responsive but less accurate, etc. The IPI times are measured with respect to the framing clock pulse, and therefore synchronization of different streams is easy. Some IPI information storage or delay may be needed but not for many time frames. This will depend on the nature of the computation needed and whether it needs inputs or results to be aligned or synchronized with other IPI inputs or results. Synchronization would be a problem for synchronous IPI signaling when the inputs have different frame times. In our work, we assume that all IPI signals in the system have the same frame time. This assumption can be made if one global clock is used to synchronize sampling and framing of all inputs. This is not a likely possibility with a network of remote sensors, and therefore a more sophisticated clock recovery and synchronization scheme such as a phase-locked loop (PLL) [120] is needed.

One advantage of the asynchronous scheme however is that it has higher bandwidth, and can potentially be more responsive than the synchronous scheme since it does not allocate a fixed-width time slot (frame) for each IPI regardless of the encoded value. Another advantage is that if a pulse is lost or a spurious pulse is received then there will be a maximum of two errors. However, with synchronous signaling, the synchronization circuitry may get stuck in an incorrect phase causing all subsequent IPIs to be erroneous. Therefore, to improve the reliability of the synchronous signaling, a more sophisticated clock-recovery scheme such as a phase-locked loop (PLL) is needed. Furthermore, a "resynch" capability is almost mandatory in any synchronous IPI representation. Despite these advantages of the asynchronous scheme, we have decided to choose the synchronous scheme for the reasons discussed above.

3.5. IPI Signal Timing Requirements

We now discuss the analog-to-IPI conversion and computation timing requirements and their effects on the IPI signal timing. These requirements are based on the synchronous signaling scheme and do not depend on any particular design implementation, i.e., they will be needed regardless of the design implementation details.

3.5.1. Analog-to-IPI Conversion Timing Requirements

Figure 3.2 shows the Analog-to-IPI conversion timing requirements. The Analog-to-IPI conversion time is equal to the IPI time, i.e., the conversion starts at the beginning of the frame and ends when the second pulse of the IPI is generated.

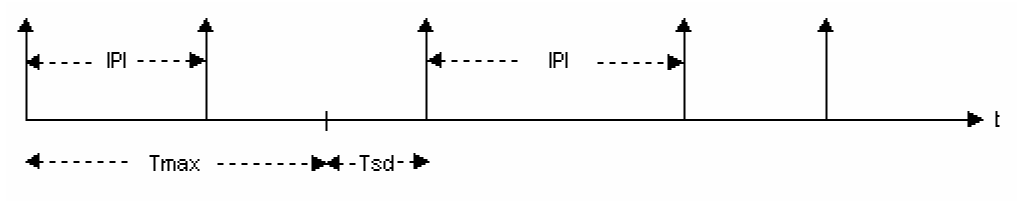


Figure 3.2. Analog-to-IPI conversion timing requirements.

To start the conversion, the analog value should be ready at the beginning of the frame. This means that the analog input should be sampled first and then converted to IPI. This also means that the sampling must begin at least T_{sd} before the frame beginning where T_{sd} is the sampling delay of the signal. To support the previous operation, we need a periodic sampling signal that has the same frequency as the framing signal but leads it by at least T_{sd}. To satisfy the Analog-to-IPI conversion timing requirements, the time frame T_f must satisfy the following two relationships:

1. $T_f > T_{sd} + T_{max}$, where T_{max} is the maximum IPI time.
2. $T_f = T_s = 1/F_s < 1/(2F_m)$, where F_s is the sampling frequency and F_m is the maximum frequency in the analog signal spectrum. According to the sampling theory, the sampling frequency F_s needs to be at least twice the maximum frequency F_m in the signal spectrum in order for the samples to completely encode all the information in the signal [105].

A possible speed optimization is to use pipelining to overlap the sampling and conversion operations so that T_{max} and T_{sd} in the first relationship above are no longer additive and T_f then needs to satisfy the following relationship: $T_f > max\ (T_{sd}\ ,\ T_{max})$.

3.5.2. *Computation Timing Requirements*

Figure 3.3 shows the computation timing requirements. Since the computation result may have an IPI shorter than the longest input IPI, the result cannot be represented in the same frame as the inputs. However, it can be represented in the next frame. Similar to the Analog-to-IPI conversion, the computation result must be ready in some form by the beginning of the next frame. The full computation can be viewed as two steps:

1. Receiving the inputs and calculating the result in some form in the current frame.
2. Converting the result from that form to IPI and generating the pulse in the next frame.

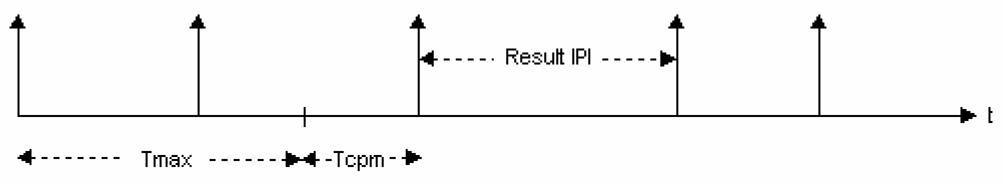


Figure 3.3. Computation timing requirements.

This is just a simple description of the basic requirements. Different computations may require different timing. For example, calculation of the result in some form may take one or more time frames. To satisfy the computation timing, the frame length T_f must satisfy the following relationship: $T_f > T_{cmp} + T_{max}$ where T_{cmp} is any computation delay that may be needed in the first frame beyond T_{max} if calculating the result in some form does not complete during T_{max}. A possible speed optimization is to use pipelining to overlap the conversion and computation operations above.

4. Novel Methods and Circuits for Time Based Conversion and Computation

In this chapter, we will present novel methods and circuits for time-based conversion and computation. These methods and circuits include IPI-to-Voltage conversion, Voltage-to-IPI conversion, and the basic arithmetic computations: addition, subtraction, division, and multiplication. These methods and circuits are based on the novel IPI representation presented in this dissertation. Therefore, their significance and novelty versus that of other pulse time representations (IPI and PWM) [69], [70], [75-81] and [98], are in many ways very similar to the significance and novelty of our IPI representation versus the other representations. The significance of these methods and circuits come also from the advantages of being able to use the same time-based representation to perform both conversion/communication and computation without the need to convert the pulse stream to and from the analog or the digital domain for computation.

Simulation results will be shown in this chapter only to demonstrate how the circuits operate. The simulation results along with the experimental test results will be presented in the next chapter. Square wave (or PWM) signals are used to convey the IPI timing information of the IPI inputs. Typically, the IPI input signal is received by a toggle flip-flop (T-FF) at the front end which generates an equivalent square wave (or PWM) signal that carries the IPI timing information. The signal is low during t_+ and high during t_-. Figure 4.1 shows how the T-FF should be connected. The intent was to focus on designing and testing the main part of each circuit and to have more direct control over the signals controlling the switching logic for better testability.

4.1. IPI-to-Voltage and Voltage-to-IPI Conversions

The relationship between T_f, t_+, and t_- is governed by the following equation

$$t_+ = T_f - t_- \tag{4.1}$$

Figure 4.1 shows the IPI-to-V-to-IPI conversion circuit. Figure 4.2 is a simulation chart that shows its operation. It takes two time frames to perform both conversions. It performs the IPI-to-V conversion in the first frame and the V-to-IPI conversion in the second frame. Initially, the capacitor voltage V_C is at V_{middle} which is the zero reference. In the first frame, each transistor is in saturation mode for its corresponding IPI time. MP1 is turned on for t_+ and MN1 is turned on for t_-. The current I in the P-channel MOS transistor should be equal in amount but opposite in direction to the current in the N-channel MOS transistor. The idea here is to linearly charge the capacitor in the positive direction during the IPI time t_+ and to linearly charge it in the negative direction at the same rate during the IPI time t_-. At the end of the first frame the output voltage on the capacitor is

$$V_C = \frac{I}{C}\left(t_+ - t_-\right) \tag{4.2}$$

In the second frame, the input transistors are turned off and MN3 and MN4 transistors are turned on in saturation mode to linearly decrease the output voltage toward $-V_M$ where $-V_M$ is the minimum output voltage (i.e., the lower limit of the output dynamic range). V_M is the maximum output voltage (i.e., the upper limit of the output dynamic range) and is equal to

$$V_M = \frac{I}{C} T_f \tag{4.3}$$

MN3 and MN4 transistors should provide the same amount of current I each. Therefore, the output voltage on the capacitor in this frame follows the following equation

$$V_C(t) = \frac{I}{C}(t_+ - t_-) - \frac{2I}{C}t \qquad (4.4)$$

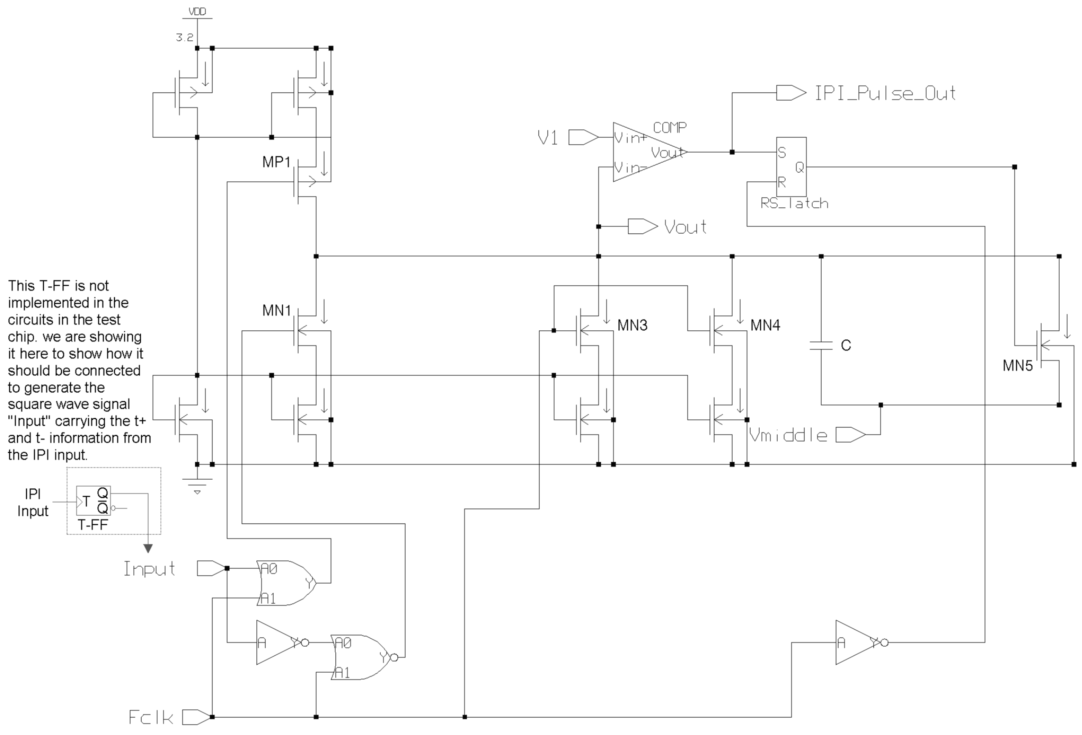

VDD
3.2
MP1
MN1
MN3
MN4
C
MN5
This T-FF is not implemented in the circuits in the test chip. we are showing it here to show how it should be connected to generate the square wave signal "Input" carrying the t+ and t- information from the IPI input.
IPI Input
T Q Q
T-FF
Input
A0
A1
Y
A
Fclk
V1
Vin+
Vin-
COMP
Vout
IPI_Pulse_Out
S
Q
R
RS_latch
Vout
Vmiddle

Figure 4.1. IPI-to-V-to-IPI conversion circuit.

When V_C reaches just below $-V_M$, the comparator output goes high. This sets the RS latch and causes its Q output to go high as well. The Q output then turns the MN5 switch on, which in turn charges the capacitor back up to its initial voltage V_{middle}. After that, the circuit is again in its initial state and is ready to start a new cycle of conversion. When V_C increases to just above $-V_M$ in its way back up to V_{middle}, the comparator output is unasserted (low), which determines the shape of the output pulse.

Some capacitance can be used to delay the rise of the Q output to adjust the width of the IPI output pulse, where a larger capacitance gives wider pulse. This extra capacitance does not affect the time at which the pulse is fired because this is decided mainly by the comparator. The pulse is generated when V_C decreases down to $-V_M$. Using (4.3) and (4.4), we can find that this happens at time

$$t_{out+} = t_+ \tag{4.5}$$

and this recovers the IPI times of the original input.

The IPI-to-V conversion circuit consumes the silicon area of the following devices:

- 1 capacitor
- 1 NMOS and 1 PMOS transistor for the current source
- 1 big NMOS discharge transistor
- 2 NMOS transistors per input
- 2 PMOS transistors per input
- 1 NOR gate, 1 OR gate, 1 inverter per input
- 1 T-FF per IPI input to generate from the IPI input signal the square wave signals that controls the input transistors

And the V-to-IPI conversion circuit costs the following devices:

- 1 capacitor (shared with IPI-to-V conversion in IPI-to-V-to-IPI conversion)
- 4 NMOS transistors which decrease the capacitor voltage towards $-V_M$ at twice the rate of IPI-to-V conversion
- 1 big NMOS discharge transistor
- 1 comparator
- 1 RS latch

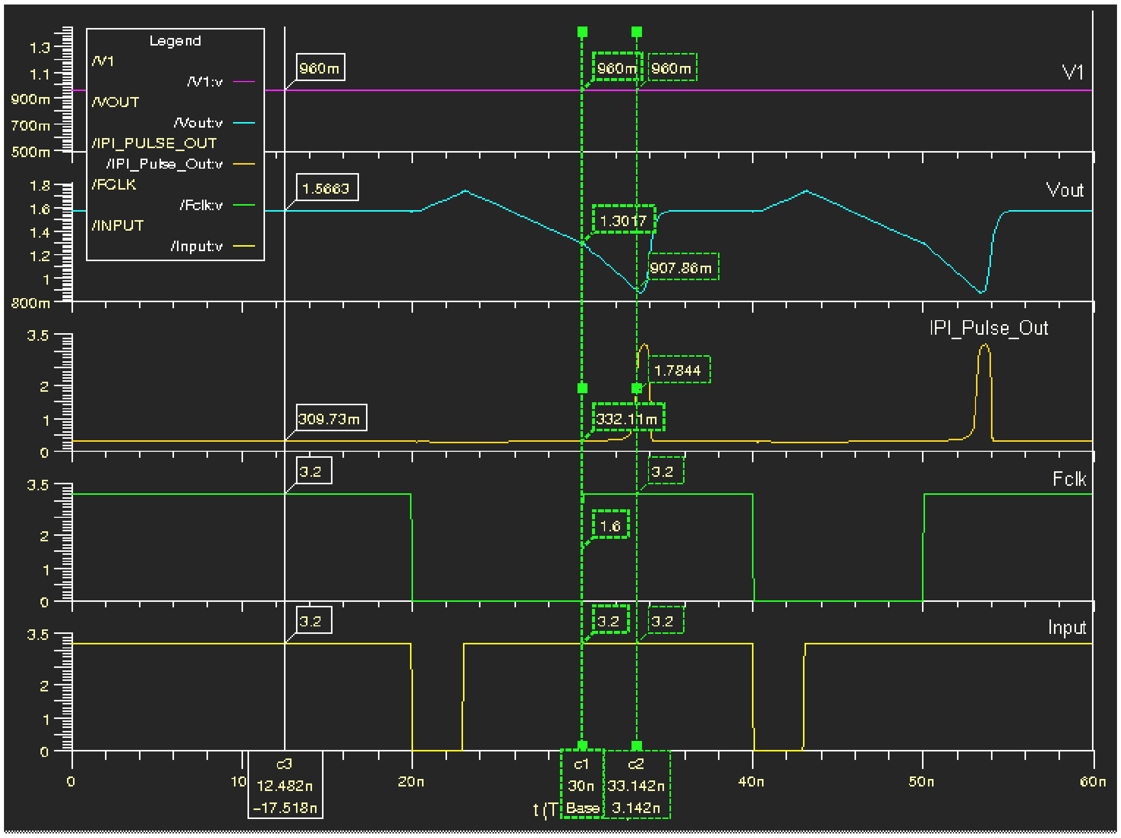


Figure 4.2. Simulation of the IPI-to-V-to-IPI conversion circuit.

Before we compare our conversion (modulation and demodulation) techniques and circuits to the other pulse time (IPI and PWM) techniques and circuits in [69], [70], [75-81] and [98], which we have reviewed in section 2.2, we would like to say that any of them can be used for conversion. However, the benefits of our representation and conversion techniques lie actually in their use for computation of negative, zero, and positive values, as we will see when we present the addition, subtraction, division, and multiplication. When compared to the other pulse time modulation (IPI and PWM) techniques and circuits in [69], [70], [75-81] and [98], our Voltage-to-IPI converter (modulator) stores the voltage sample on the capacitor. Then, as we explained above, it charges the capacitor linearly down toward $-V_M$ at twice the charging rate of the IPI-to-V conversion. When equivalence is detected, the comparator generates the pulse. The other techniques, however, do not ramp the input voltage on the capacitor. Instead, a sawtooth ramp signal is provided and compared against the input sample for equivalence. The ramp itself and/or a DC offset can be used to represent negative values. When compared to the other pulse time demodulation (IPI and PWM) techniques and circuits in [69], [70], [75-81] and [98], our IPI-to-voltage converter (demodulator), as we described above, charges the capacitor in the positive direction during the first part of the frame t_+, and in the negative direction during the second part of the frame t_-. This basically realizes (4.2), which is a mathematical description of the novel IPI representation presented in section 3.1. In the other techniques, however, PWM signals are converted to IPI signals first. Then, IPI demodulation is performed. For IPI demodulation, the first input pulse is used to initiate the generation of a sawtooth ramp signal and the second input pulse is used to sample and hold the ramp signal. This produces the voltage sample equivalent to the IPI.

4.2. Addition

Figure 4.3 shows the IPI addition circuit. Figure 4.4 is a simulation chart that shows its operation and Figure 4.5 is a timing diagram that shows the phases and sequence of computation. Its operation is very similar to the IPI-to-V-to-IPI conversion except for having a second input. Initially the capacitor voltage V_C is at V_{middle}. In the first frame, each input transistor is turned on in saturation mode for its corresponding IPI time. MP1 is turned on for t_{1+} and MN1 is turned on for t_{1-}. Similarly, MP2 is turned on for t_{2+} and MN2 is turned on for t_{2-}. Therefore, each input transistor contributes linearly to the output voltage on the shared capacitor and positively or negatively according to its t_+ and t_- times. At the end of the first frame the output voltage on the capacitor is

$$V_C = \frac{I}{C}\left[(t_{1+} - t_{1-}) + (t_{2+} - t_{2-})\right] \qquad (4.6)$$

In the second frame, the operation is identical to the voltage-to-IPI conversion explained in the previous section. All the input transistors are turned off and MN3 and MN4 transistors are in saturation mode to linearly decrease the output voltage toward $-V_M$. In this frame, the output voltage on the capacitor follows the following equation

$$V_C(t) = \frac{I}{C}\left[(t_{1+} - t_{1-}) + (t_{2+} - t_{2-})\right] - \frac{2I}{C}t \qquad (4.7)$$

The output pulse is generated when V_C decreases to $-V_M$. Using (4.3) and (4.7), we can find that this happens at time

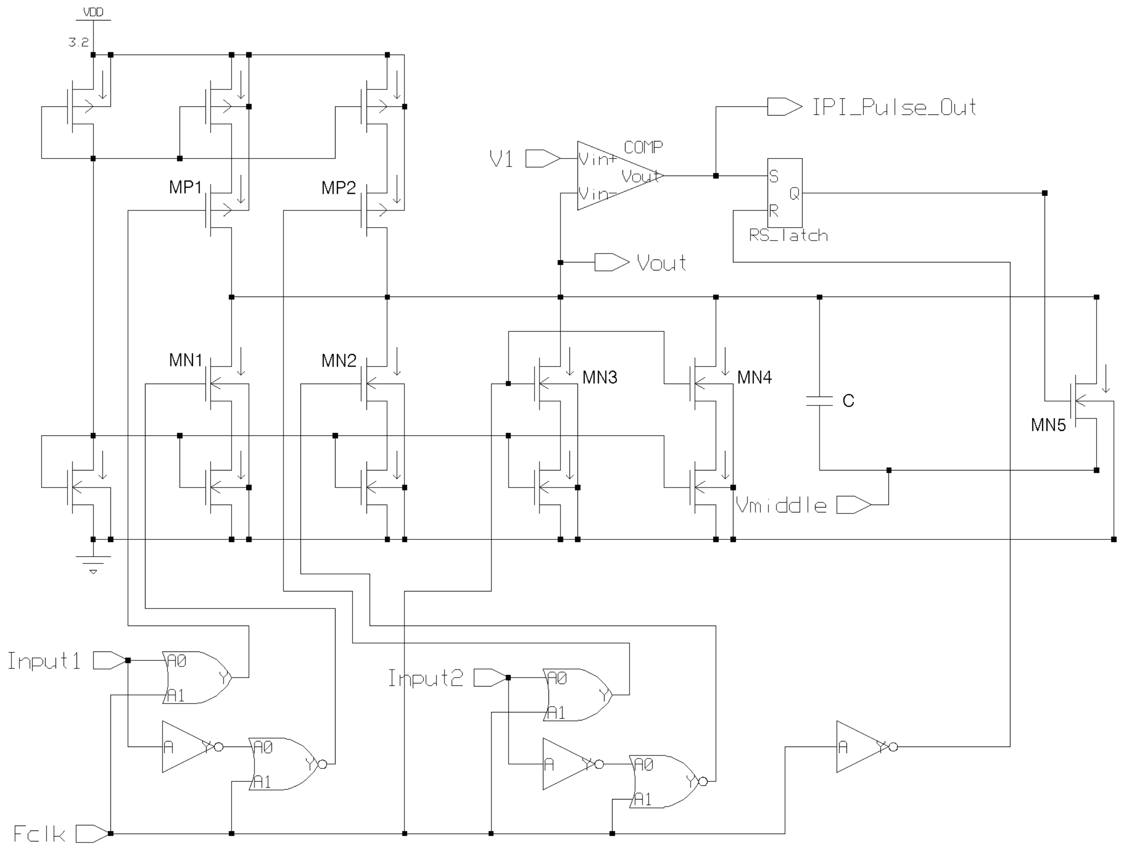


Figure 4.3. IPI addition circuit.

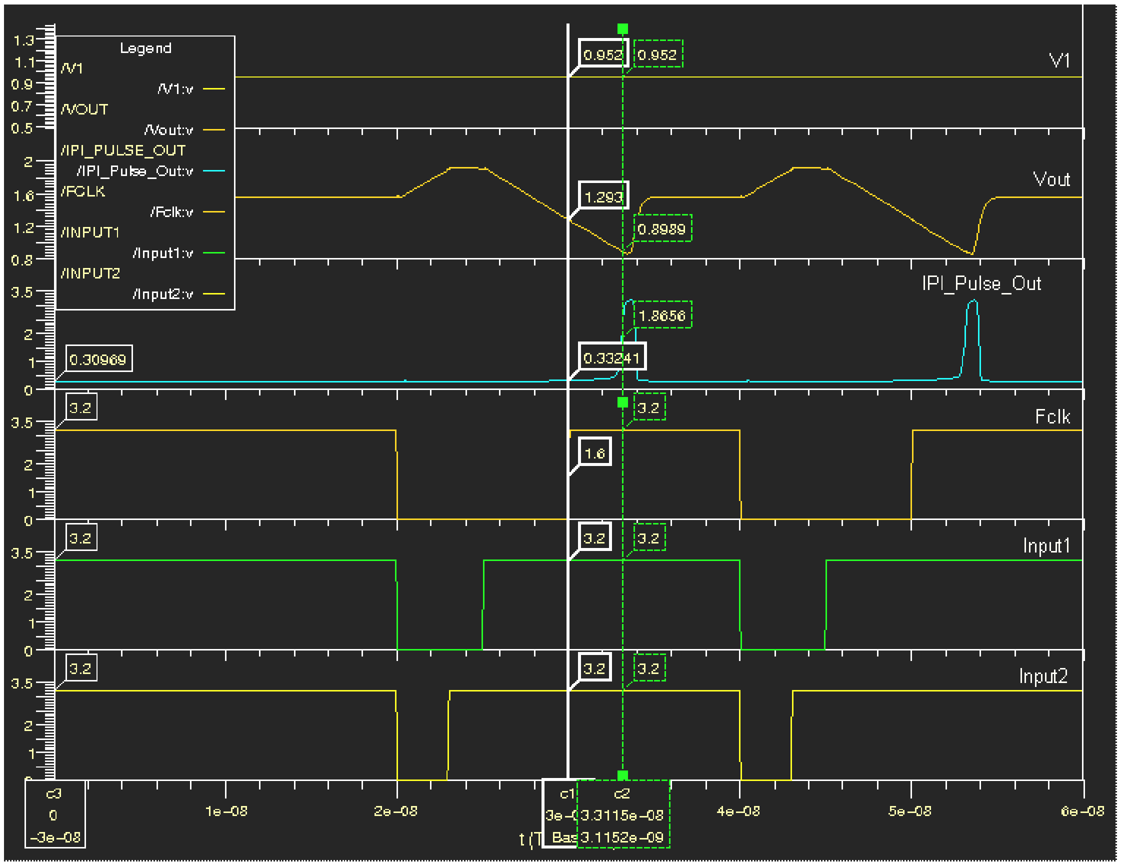
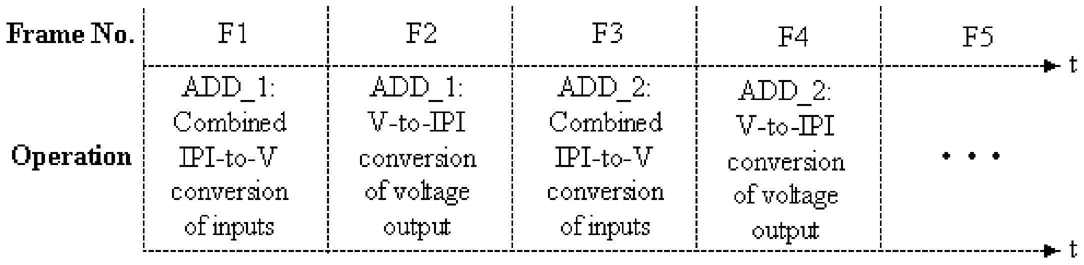


Figure 4.4. Simulation of the IPI addition circuit.



Frame No.	F1	F2	F3	F4	F5
Operation	ADD_1: Combined IPI-to-V conversion of inputs	ADD_1: V-to-IPI conversion of voltage output	ADD_2: Combined IPI-to-V conversion of inputs	ADD_2: V-to-IPI conversion of voltage output	• • •

Figure 4.5. Timing diagram of addition.

$$t_{3+} = \left(t_{1+} + t_{2+}\right) - \frac{T_f}{2} \qquad (4.8)$$

and using (4.1) and (4.8), we can find

$$t_{3+} - t_{3-} = \left(t_{1+} - t_{1+}\right) + \left(t_{2+} - t_{2-}\right) \qquad (4.9)$$

The addition circuit consumes the silicon area of the following devices:

- 1 capacitor
- 1 NMOS and 1 PMOS transistor for the current source
- 1 big NMOS discharge transistor
- 4 NMOS transistors which decrease the capacitor voltage towards $-V_M$ at twice the rate of IPI-to-V conversion
- 1 comparator
- 1 RS latch
- 2 NMOS transistors per input
- 2 PMOS transistors per input
- 1 NOR gate, 1 OR gate, 1 inverter per input
- 1 T-FF per IPI input to generate from the IPI input signal the square wave signals that controls the input transistors

While it is true that we can use the pulse time (IPI and PWM) representation techniques, which we have reviewed in section 2.2, for conversion/communication, their use for direct “time-based” computation has not been demonstrated anywhere, and is difficult, as we explained in section 3.1. Analog computation can be used with these representations, after converting the pulse streams to analog, but the thesis will show that analog computation suffers from problems that our circuits do not have such as the body effect and the mobility degradation effect. In this section, we have demonstrated how to use our representation approach and IPI-to-V and V-to-IPI conversion techniques, to perform addition of negative, zero, and positive values. All the inputs are IPI and the output is also IPI and is ready for use by other IPI computation blocks. This discussion is also valid for all the other computation circuits, which we discuss next.

4.3. Subtraction

The subtraction circuit is the same as the addition circuit in Figure 4.3 except for the second input is inverted so that its logic and transistors operate in an opposite manner. MN2 is turned on during t_{2+} instead of MP2 and MP2 is turned on during t_{2-} instead of MN2. This way the second input is subtracted from the first input not added to it. Figure 4.6 is a simulation chart that shows the subtraction operation. At the end of the first frame the output voltage on the capacitor is

$$V_C = \frac{I}{C}\left[(t_{1+} - t_{1-}) - (t_{2+} - t_{2-})\right] \tag{4.10}$$

In the second frame, the output voltage on the capacitor follows the following equation

$$V_C(t) = \frac{I}{C}\left[(t_{1+} - t_{1-}) - (t_{2+} - t_{2-})\right] - \frac{2I}{C}t \tag{4.11}$$

The output pulse is generated when V_C decreases to $-V_M$. Using (4.3) and (4.11), we can find that this happens at time

$$t_{3+} = (t_{1+} - t_{2+}) + \frac{T_f}{2} \tag{4.12}$$

and using (4.1) and (4.12), we can find

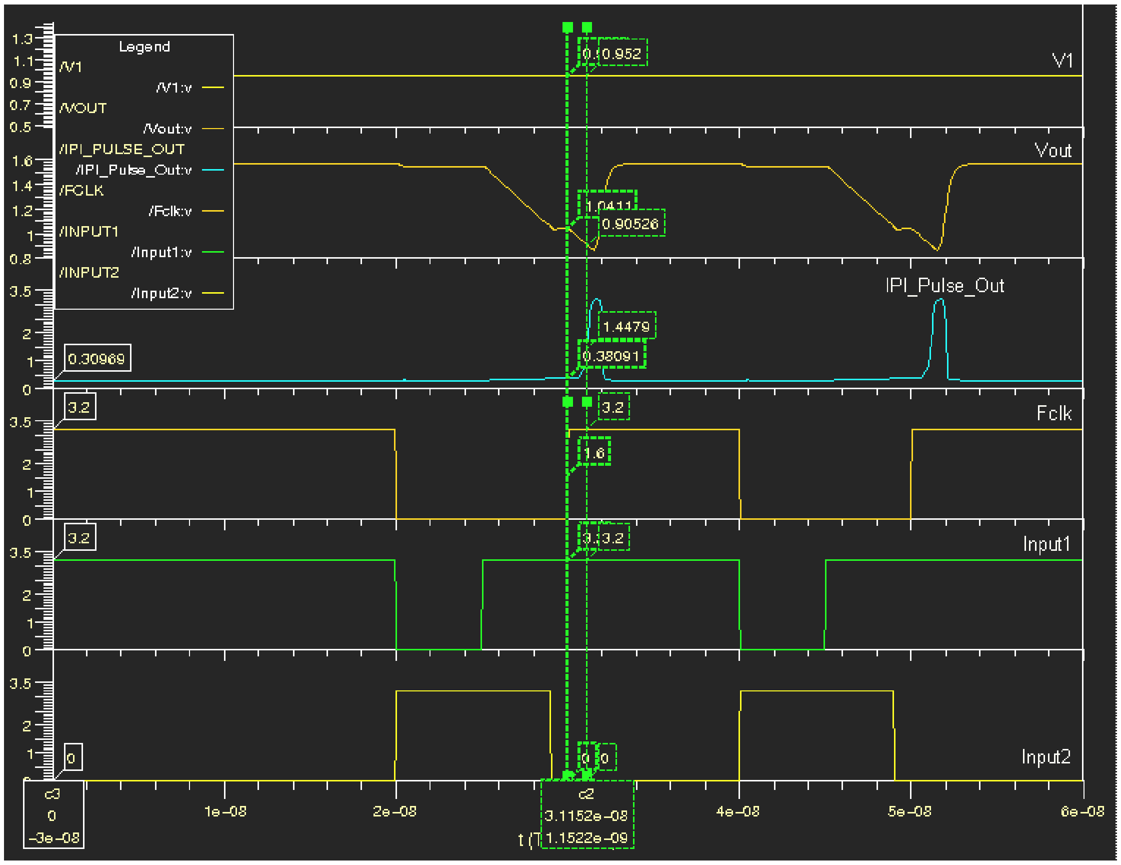


Figure 4.6. Simulation of the IPI subtraction circuit.

$$t_{3+} - t_{3-} = (t_{1+} - t_{1+}) - (t_{2+} - t_{2-}) \qquad (4.13)$$

4.4. Division

Figure 4.7 shows the time-based division circuit. Figure 4.8 is a simulation chart that shows its operation and Figure 4.9 is a timing diagram that shows the phases and sequence of computation. This division circuit, which is also used for multiplication in the next section, operates in the first quarter but it is the core of the four-quadrant (4-Q) operation. In section 4.6, we will explain how it can be used for 4-Q operation with minimum modifications.

Initially, all capacitors in the circuit are at V_{middle}. In the first frame, each input transistor is turned on in saturation mode for its corresponding IPI time. MP1 is turned on for t_{1+} and MN1 is turned on for t_{1-}. Similarly, MP2 is turned on for t_{2+} and MN2 is turned on for t_{2-}. At the end of the first frame, the voltages on the capacitors C_1 and C_2 are

$$V_{C1} = \frac{I_1}{C_1}\left(t_{1+} - t_{1-}\right) \qquad (4.14)$$

$$V_{C2} = \frac{I_2}{C_2}\left(t_{2+} - t_{2-}\right) \qquad (4.15)$$

V_{C1} is connected to the positive input of the voltage follower OP-AMP. This sets the voltage across the resistor R to V_{C1} and therefore the current in R is

$$I_R = \frac{V_{C1}}{R} \qquad (4.16)$$

The current in M3 and M4 is equal to I_R. In the second half of the second frame, M5 is turned on and therefore the current charging the capacitor C_3 is

$$I_{C3} = I_{M5} = I_{M6} = \frac{\left(W_6/L_6\right)}{\left(W_4/L_4\right)} I_{M4} = \frac{\left(W_6/L_6\right)}{\left(W_4/L_4\right)} I_R \qquad (4.17)$$

where W_4, W_6, L_4, and L_6 are the channel widths and lengths of MN4 and MN6, respectively. Consequently, V_{C3} will increase linearly according to the following equation

$$V_{C3} = \frac{I_{C3}}{C_3}\left(t - \frac{T_f}{2}\right) \qquad (4.18)$$

When V_{C3} reaches a level equivalent to V_{C2} the comparator detects that situation and the output pulse is generated. Using (4.14)-(4.18), we can find that the time when this happens is

$$t_{3+} = \left(\frac{I_{C2}}{I_{C1}}\right)\left(\frac{RC_1C_3}{C_2}\right)\frac{(W_6/L_6)}{(W_4/L_4)}\frac{(t_{2+}-t_{2-})}{(t_{1+}-t_{1-})}+\frac{T_f}{2} \qquad (4.19)$$

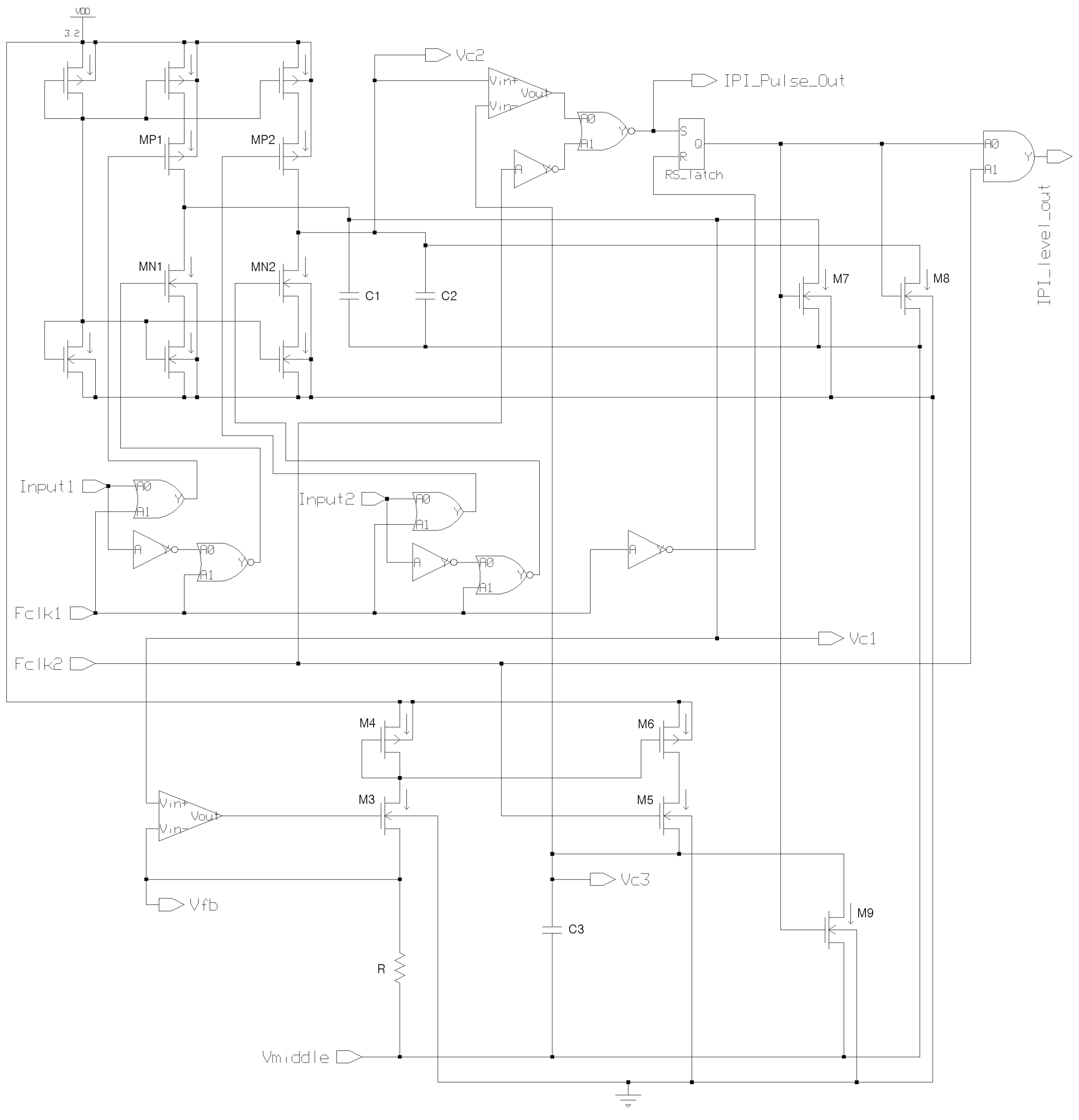


Figure 4.7. IPI division circuit.

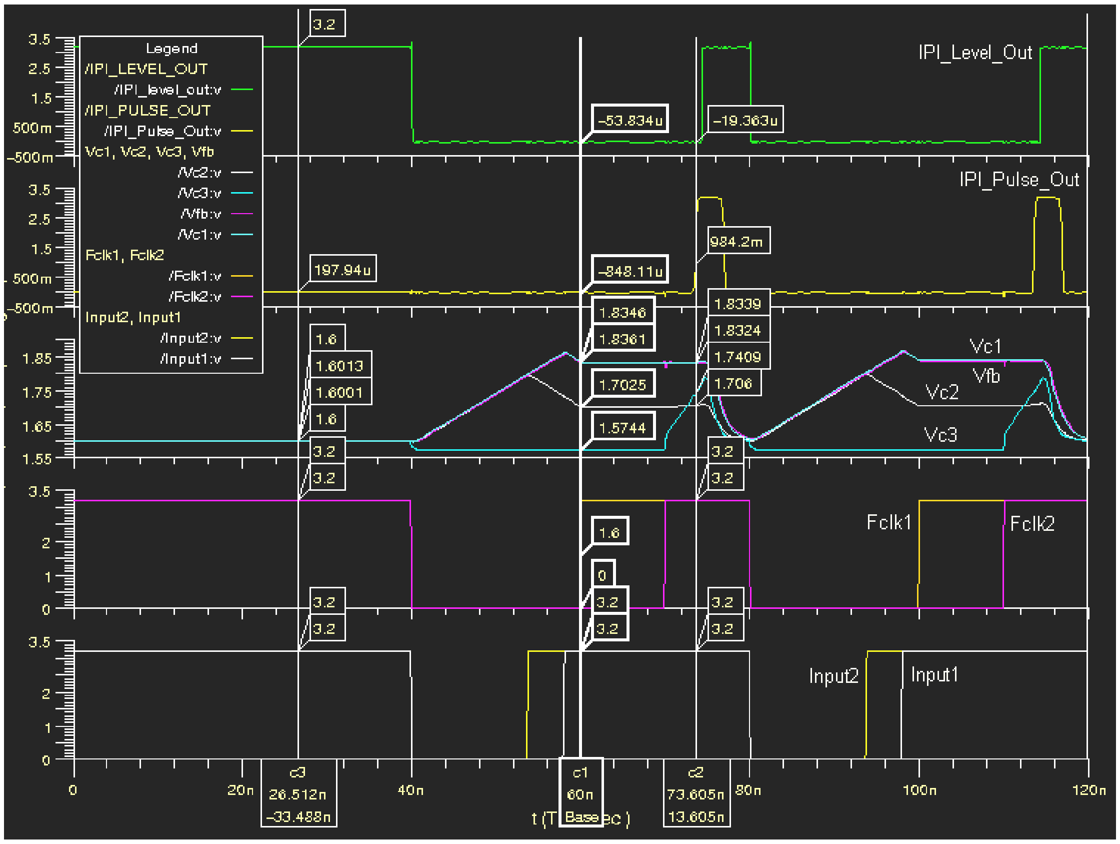
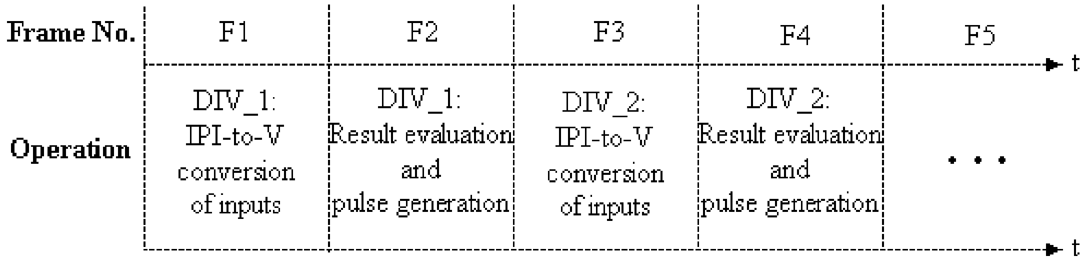


Figure 4.8. Simulation of the IPI division circuit.



Frame No.
F1
F2
F3
F4
F5
t
Operation
DIV_1: IPI-to-V conversion of inputs
DIV_1: Result evaluation and pulse generation
DIV_2: IPI-to-V conversion of inputs
DIV_2: Result evaluation and pulse generation
. . .
t

Figure 4.9. Timing diagram of division.

and using (4.1) and (4.19),

$$t_{3+} - t_{3-} = K_{div} \frac{(t_{2+} - t_{2-})}{(t_{1+} - t_{1-})} \tag{4.20}$$

where

$$K_{div} = 2\left(\frac{I_{C2}}{I_{C1}}\right)\left(\frac{RC_1C_3}{C_2}\right)\frac{(W_6/L_6)}{(W_4/L_4)} \tag{4.21}$$

As we can see from (4.20), the output represented by the IPI times t_{3+} and t_{3-} is a scaled value of the division of the second input represented by the IPI times t_{2+} and t_{2-}, by the first input represented by the IPI times t_{1+} and t_{1-}. The scaling factor K_{div} can be set by choosing the right values for the design parameters I_{C1}, I_{C2}, R, C_1, C_2, C_3, W_4, W_6, L_4, and L_6 based on the design requirements such as power, area, speed, and accuracy.

The division circuit has the following silicon cost:

- 3 capacitors
- 1 resistor
- 1 NMOS and 1 PMOS transistor for the current source
- 3 big NMOS discharge transistors per capacitor
- 1 comparator
- 1 OP-AMP
- 1 RS latch
- 1 NOR gate and 1 AND gate
- 2 NMOS and 2 PMOS to mirror the current in the resistor to C_3
- 2 NMOS transistors per input
- 2 PMOS transistors per input
- 1 NOR gate, 1 OR gate, 1 inverter per input

- 1 T-FF per IPI input to generate from the IPI input signal the square wave signals that controls the input transistors

As the divisor gets very small or zero, the division result grows very large and the output pulse will never be generated. The capability of detecting such situation can be added to our circuit by employing a timeout/saturation mechanism. Maximum output pulse signal, say IPI_{MAX}, is derived from the framing pulse signal but it leads it by a short time t_{lead}. This short time should be long enough to accommodate the pulse width and the circuit reset time. It can be inserted using an even number of inverters. If an output pulse is generated during the output frame then there is no need for the IPI_{MAX} pulse and it should be masked by the IPI_level_out signal. On the other hand, if no output pulse has been generated then the IPI_{MAX} pulse should not be masked and should be taken as the output pulse.

4.5. Multiplication

Figure 4.10 shows a block diagram of the IPI based multiplication circuit. Figure 4.11 is a simulation chart that shows its operation and Figure 4.12 is a timing diagram that shows the phases and sequence of computation. It uses two division circuits. The idea of using division to compute the multiplication comes from the fact that multiplying by a number is equivalent to dividing by its inverse. Thus, the multiplication can be performed as two consecutive divisions as follows

$$xy = x/(1/y) \tag{4.22}$$

The first division circuit takes the first input to be multiplied as its first input and a constant signal as its second input and computes the following result in the second frame

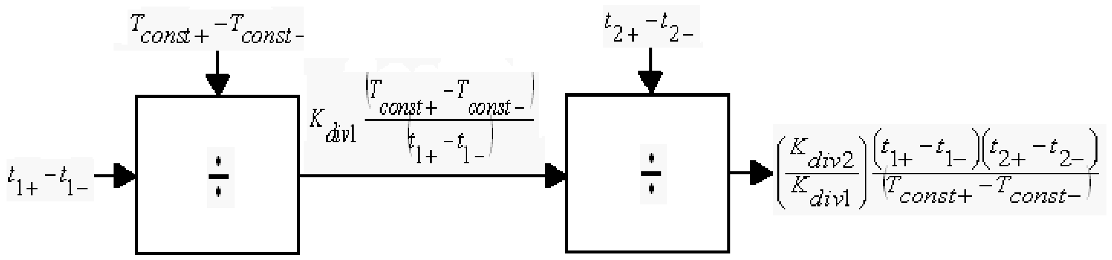

$T_{const+} - T_{const-}$
$t_{1+} - t_{1-}$
÷
$K_{div1} \frac{(T_{const+} - T_{const-})}{(t_{1+} - t_{1-})}$
$t_{2+} - t_{2-}$
÷
$\left(\frac{K_{div2}}{K_{div1}}\right) \frac{(t_{1+} - t_{1-})(t_{2+} - t_{2-})}{(T_{const+} - T_{const-})}$

Figure 4.10. IPI multiplication as two IPI divisions.

$$t_{3+} - t_{3-} = K_{div1} \frac{(T_{const+} - T_{const-})}{(t_{1+} - t_{1-})} \tag{4.23}$$

where T_{const+} and T_{const-} are the IPI times representing the constant input. The second division circuit takes the result, which the first division circuit computes, as its first input and the second input to be multiplied as its second input and computes the following result in the third frame

$$t_{4+} - t_{4-} = K_{div2} \frac{(t_{2+} - t_{2-})}{(t_{3+} - t_{3-})} \tag{4.24}$$

Using (4.23) and (4.24), we can find

$$t_{4+} - t_{4-} = K_{mult} (t_{1+} - t_{1-})(t_{2+} - t_{2-}) \tag{4.25}$$

where

$$K_{mult} = \left(\frac{K_{div2}}{K_{div1}}\right) \frac{1}{(T_{const+} - T_{const-})} \tag{4.26}$$

As we can see from (4.25), the output represented by the IPI times t_{4+} and t_{4-} is a scaled value of the multiplication of the first input represented by the IPI times t_{1+} and t_{1-}, by the second input represented by the IPI times t_{2+} and t_{2-}. The scaling factor K_{mult}

can be set by choosing the right values for K_{div1}, K_{div2}, and the constant input IPI times. From process variations point of view, this multiplication method has a great advantage since these variations in devices like transistors, capacitors, resistors will cancel out if the two division circuits are close from each other, since K_{div2} is divided by K_{div1}.

Since the multiplication uses two division circuits, it has the same problem with small or zero inputs as the division does. Therefore, the timeout/saturation capability described above for division is also needed. It also takes care of any situation in which the output pulse is not generated for any reason.

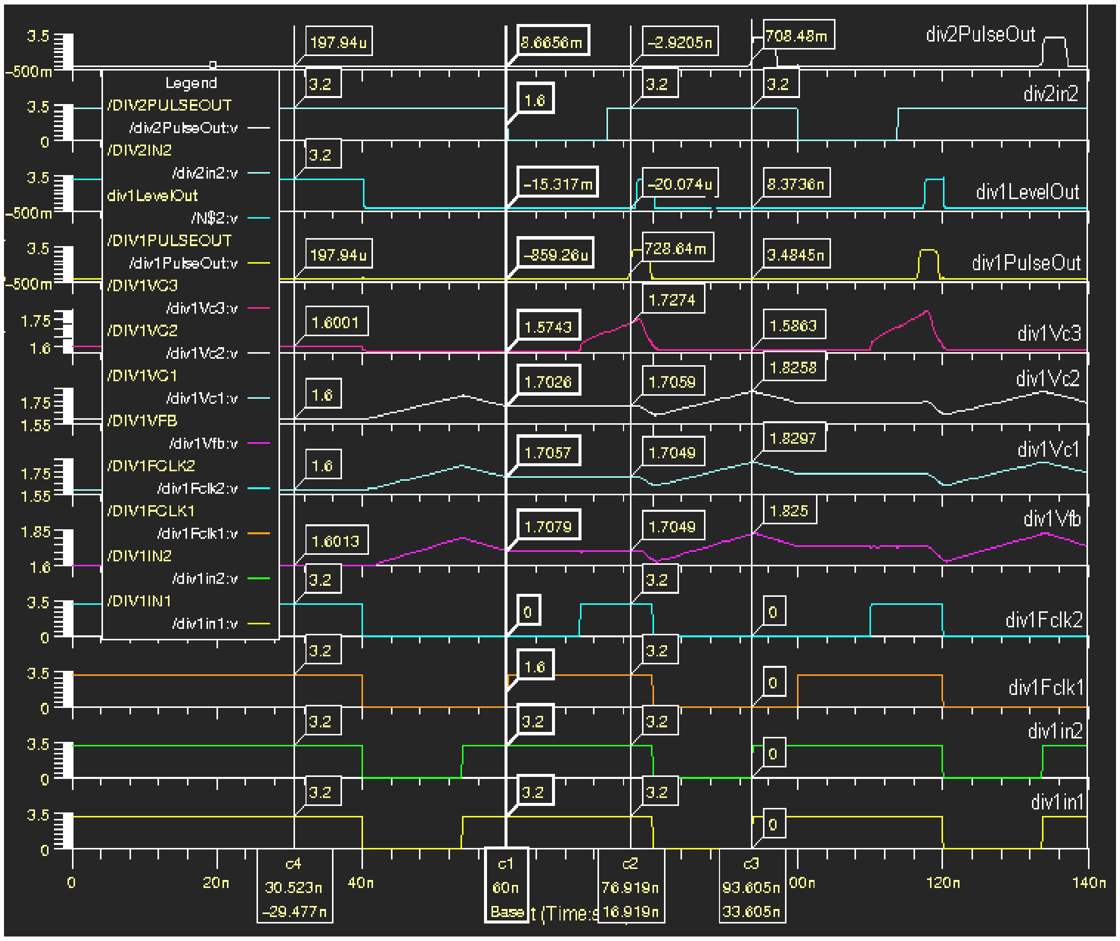


Figure 4.11. Simulation of the IPI multiplication circuit.

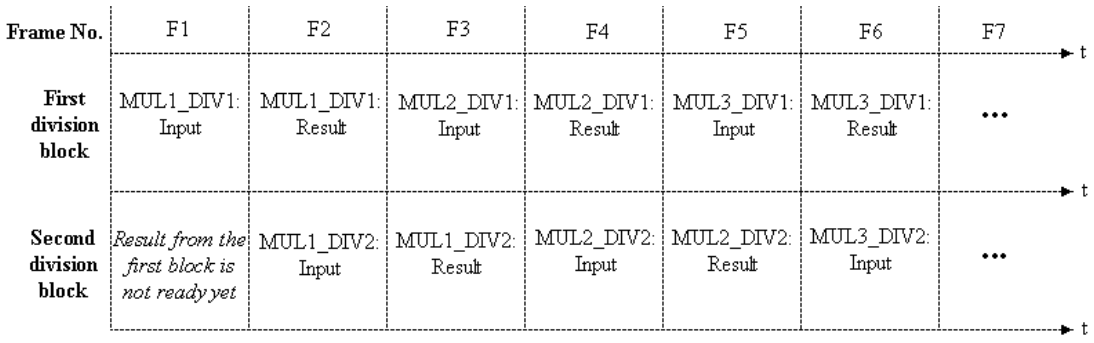


Figure 4.12. Timing diagram of multiplication.

Multiplication runs at the same speed as division, one operation per two frames. The first result requires three frames but subsequent results are available every two frames. This is because the two division blocks operate in a pipelined fashion. As the first division block is producing its result in one frame, the second division block is taking this result as input and performing the IPI-to-V conversion of the inputs in the same frame. This can be seen by looking at the timing diagram in Figure 4.12.

Unlike the pulse based multiplication techniques and circuits in [7-10] and [16-18], which we reviewed in section 2.2, the inputs to our time-based multiplier are all time-based (IPI or PWM) and the output is also time based (IPI or PWM) and is ready for use by other computation blocks or for communication. Moreover, our multiplier does not use any analog or digital multiplication techniques. In [7-10], one input is a voltage while the other can be a PFM/PWM signal and an analog transconductance multiplier is used. In [16-18], one input is digital while the other is a PFM signal and a shift register with a clock that has a varying binary-weighted cycle time (*T, 2T, 4T, 8T, ...,* $2^{N-1}T$) is used. As we explained in section 3.1, doing both conversion/communication and computation in the same time domain with no need to convert to/from analog or digital has certain advantages. Avoiding analog computation eliminates serious problems such as the body effect and the mobility degradation effect, as we will see in subsection 6.2.1. There are situations however where a designer may be willing to choose the analog computation for its simplicity and speed.

Avoiding digital computation eliminates the need for ADC, which is very expensive in terms of area and power consumption, as we will see in subsection 6.2.2.

4.6. Four-Quadrant Division and Multiplication

In this section, we will explain how the first-quadrant (1-Q) division/multiplication circuit above can be used for 4-Q operation. The discussion uses division but it also applies to multiplication.

If we have two inputs *x* and *y* then their division

$$z = x/y = \begin{cases} +|x/y| & if \quad x \geq 0 \ \ and \ \ y \geq 0 \\ -|x/y| & if \quad x \geq 0 \ \ and \ \ y < 0 \\ -|x/y| & if \quad x < 0 \ \ and \ \ y \geq 0 \\ +|x/y| & if \quad x < 0 \ \ and \ \ y < 0 \end{cases} \tag{4.27}$$

This means that we only need to compute two results $+|x/y|$ and $-|x/y|$. Then, based on the signs of the two inputs, we select one of them as the final result. Our IPI circuit has the result in PWM form (square wave form). Obtaining $-|x/y|$ from $+|x/y|$ can be done simply using an inverter. The delay of the inverter has negligible impact on the accuracy since its low-to-high and high-to-low propagation delays, t_{pLH} and t_{pHL}, are practically equal and therefore will have negligible impact on the high and low durations of the inverted output. The result in IPI form can be easily obtained from the PWM result if needed. In light of this, our problem has been reduced to calculating only one value $+|x/y|$ whatever the signs of the two inputs are, obtaining $-|x/y|$ using an inverter, and then selecting one of them as the final result based on the signs using a 2-by-1 multiplexer (MUX).

Our 1-Q circuit computes the result $+|x/y|$. The voltages V_{C1} and V_{C2} are ready by the end of the first frame. Therefore, the signs of the two inputs can be generated at the end of the first frame by using two comparators with their negative terminals connected to V_{middle}. If the voltage is above V_{middle} then it is positive and if it is below V_{middle} then it is negative. The sign of the final result can be generated from the signs of the two inputs using a logical XOR gate, and it is used to control the 2-by-1 MUX. If the two signs are similar then $+|xy|$ will be selected as the final result and if they are different then $-|xy|$ will be selected. Our 1-Q circuit works for positive values of V_{C1} and V_{C2}. V_{C1} is connected to the positive terminal of the feedback voltage follower OP-AMP and V_{C2} is connected to the positive terminal of the comparator which compares V_{C3} against V_{C2}. We need to provide the voltage follower OP-AMP and the comparator with the absolute values of V_{C1} and V_{C2} ($|V_{C1}|$ and $|V_{C2}|$), respectively. To do this, we need to generate $-V_{C1}$ and $-V_{C2}$ using IPI-to-V converters but with the PWM inputs inverted. Full IPI-to-V converters are not needed. Only the switches, current sources, and the capacitor are needed. The PWM signals controlling the switches are available from the already existing IPI-to-V converters used to generate V_{C1} or V_{C2}. They should be swapped though to negate the inputs. Then, the value or its negative is passed to the OP-AMP or the comparator using a pass gate controlled by its sign. If the value is positive then it will pass. If it is negative then its absolute value will be passed. This will ensure that the OP-AMP and the comparator always get the absolute values of V_{C1} and V_{C2}, respectively.

As we can see from the discussion above, the 1-Q circuit is the core of the 4-Q operation and it only needs the following to operate as 4-Q:

- 1 inverter to obtain $-|xy|$ from $+|xy|$
- 2 comparators to generate the signs of the two inputs
- 1 XOR gate to generate the sign of the final result and select $+|xy|$ or $-|xy|$
- 1 2-by-1 MUX
- 2 IPI-to-V converters to negate V_{C1} and V_{C2}. T-FFs and input logic gates are not necessary since signals controlling the switches are available from the other IPI-to-V converters generating V_{C1} and V_{C2}.
- 2 pass gates to pass V_{C1} or $-V_{C1}$ to the voltage-follower OP-AMP

- 2 pass gates to pass V_{C2} or $-V_{C2}$ to the comparator

Based on our 0.35 um implementation of the 1-Q circuit in chapter 5, this extra circuitry will increase the area by about 60% and the power consumption by about 45%. The 4-Q circuit does not require any extra time frames and therefore runs at the same speed as the 1-Q circuit. There are delays added by the inverter and the MUX in the path of the PWM output signal but these delays are usually very small compared to the time frame. Also, as we explained above, their propagation delays have negligible impact on the high and low durations of the PWM output signal since the low-to-high delay practically cancels the high-to-low delay. Furthermore, the parasitic capacitances of the pass gates are negligible if the capacitors are made large enough. Therefore, the accuracy of the 4-Q circuit is very similar to that of its core, the 1-Q circuit.

5. Experimental Results

We have implemented the methods and circuits described in the previous chapter in a chip using the TSMC 0.35 um mixed-signal CMOS fabrication process technology thorough the MOSIS-USA fabrication and packaging service [118] and [119]. The process has four metal layers and two poly Silicon layers. In this chapter, we describe the design of the chip including the top level cells. We have simulated and tested the circuits at a 10 MHz framing speed which corresponds to a 100 ns time frame. We have also simulated the conversion, addition, and subtraction circuits at 100 MHz and the division and multiplication circuits at 50 MHz. We present the experimental test and simulation results for each of the conversion and computation circuits in terms of accuracy, area, power consumption, and dynamic range.

5.1. Top Level Chip Design

The chip design, layout, simulation, and verification were performed using the Mentor Graphics ASICs Design Kit (ADK) and tools [116]. The chip has one addition circuit that can also be used to demonstrate subtraction and IPI-to-V-to-IPI conversion. The chip also has two division circuits to demonstrate division and multiplication. Figures 5.1-5.5 show the layout design of the OP-AMP/comparator, RS latch, addition/subtraction/IPI-V-IPI conversion, division, and top level cells, respectively. The OP-AMP/comparator and RS latch cells are lower level cells used by the addition and division cells. Figure 5.6 is a microphotograph of the chip. The total chip area including the input and output pads is 2.25 mm^2 (1.5 mm × 1.5 mm).

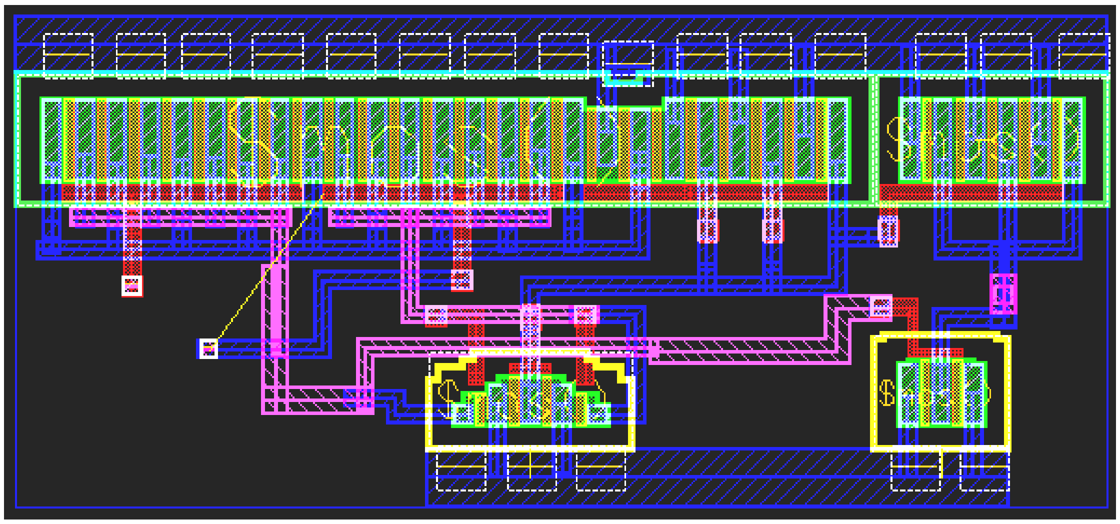


Figure 5.1. Layout of the OP-AMP/comparator cell. The cell was custom designed since the ADK library does not have OP-AMP or comparator cells.

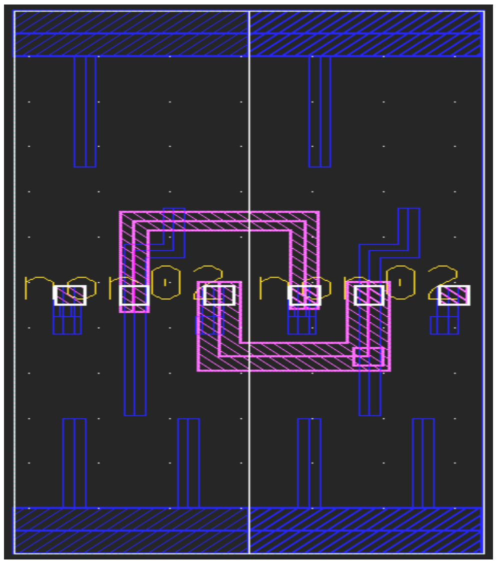


Figure 5.2. Layout of the RS latch cell. Two standard NOR gate cells from the ADK library were used to construct this cell.

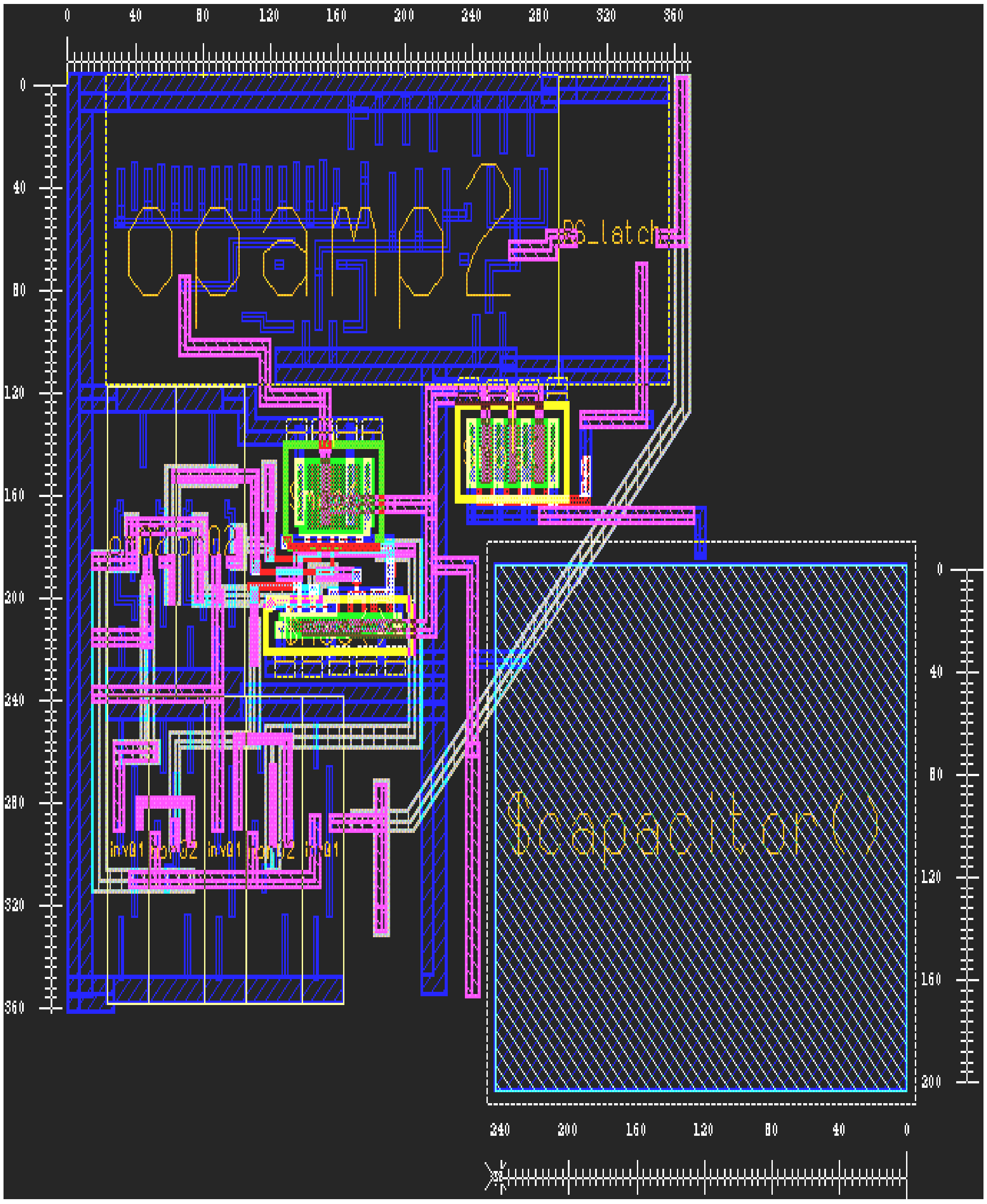


Figure 5.3. Layout of the addition, subtraction, and IPI-V-IPI conversion cell. The area is about 480λ × 400λ, which is equal to 7680 um^2 (λ = 0.2 um for 0.35 um process).

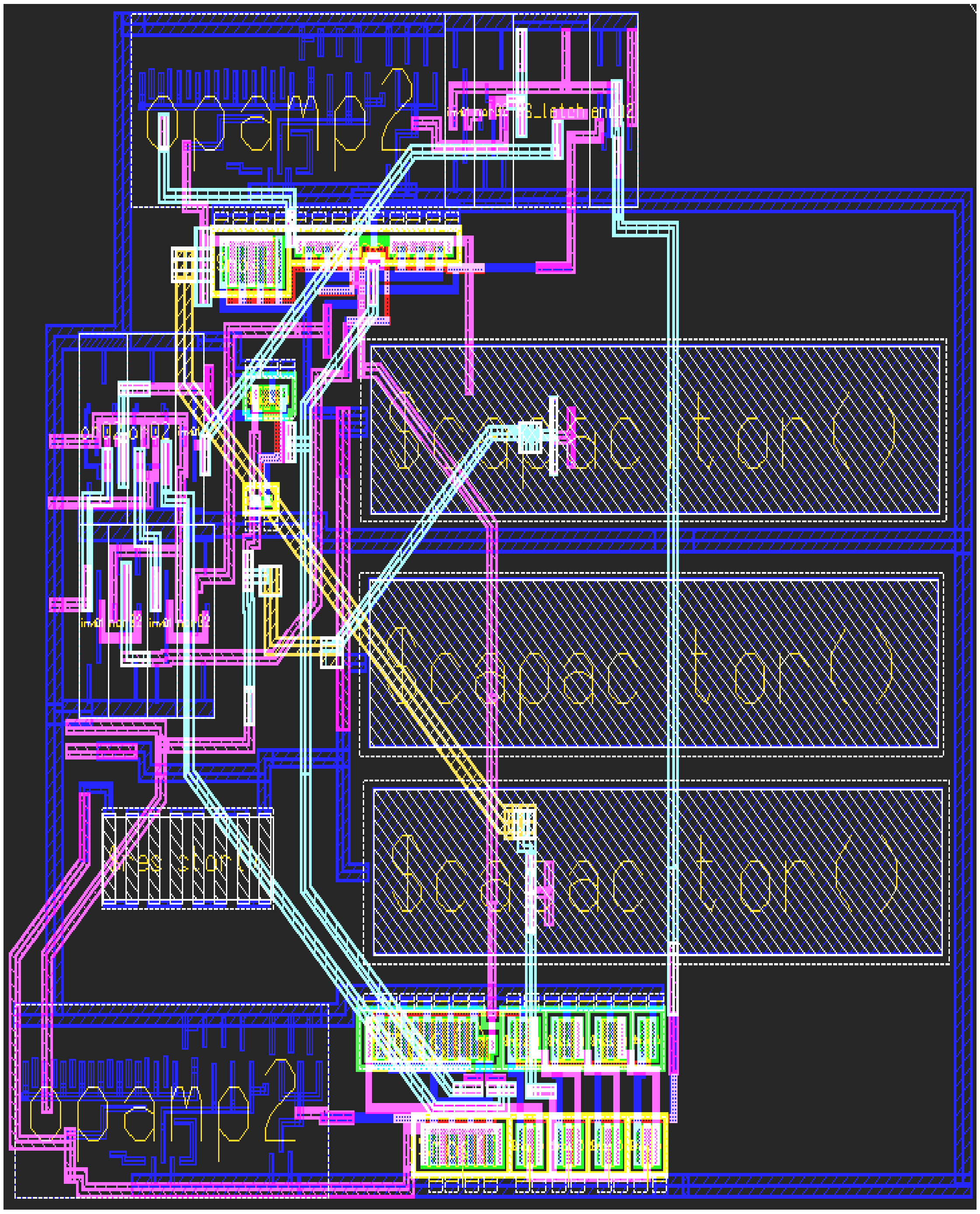


Figure 5.4. Layout of the division cell. The area is about 900λ × 720λ, which is equal to 25920 um^2 (λ = 0.2 um for 0.35 um process).

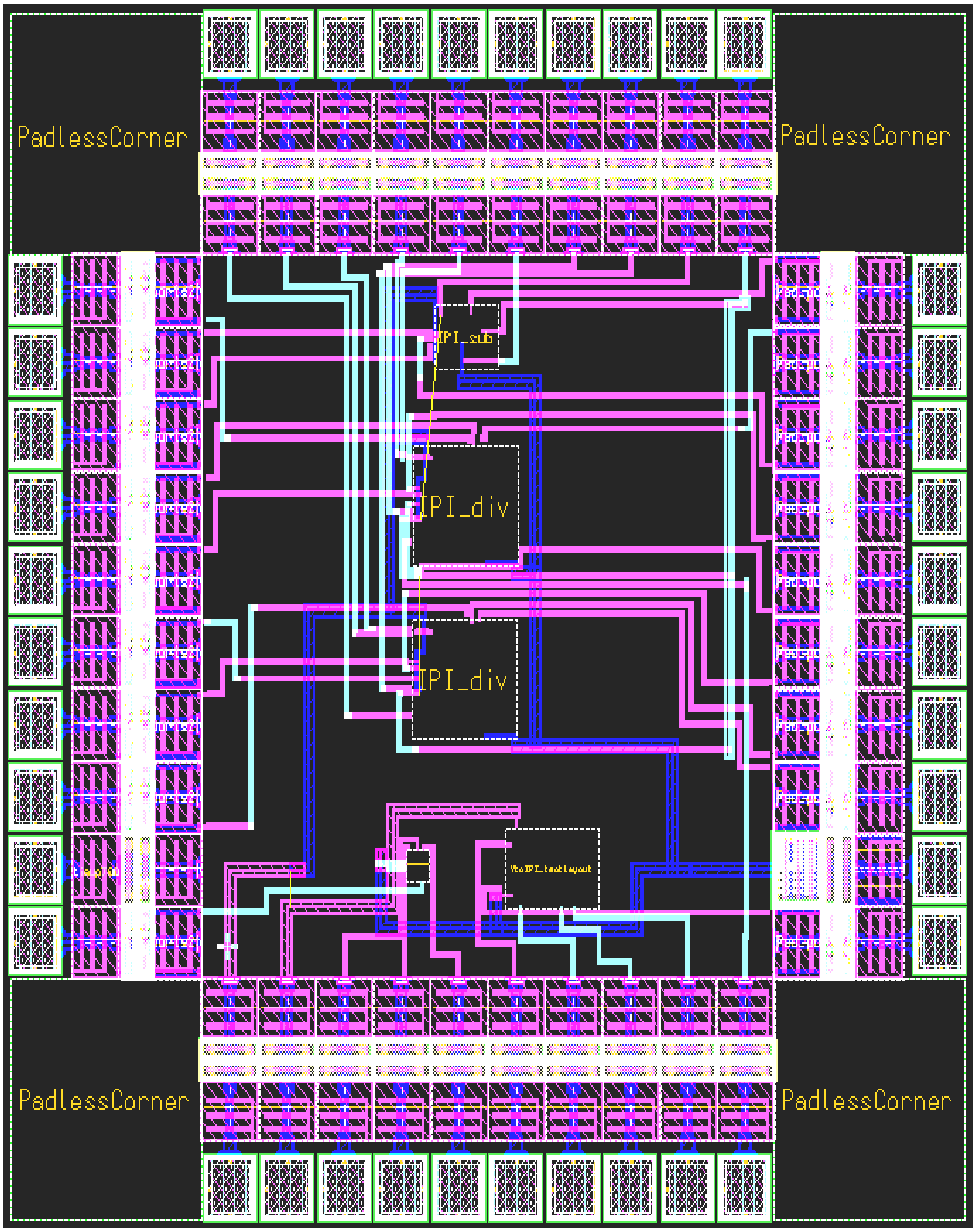


Figure 5.5. Top-level chip layout. The area is 1.5 mm × 1.5 mm = 2.25 mm^2.

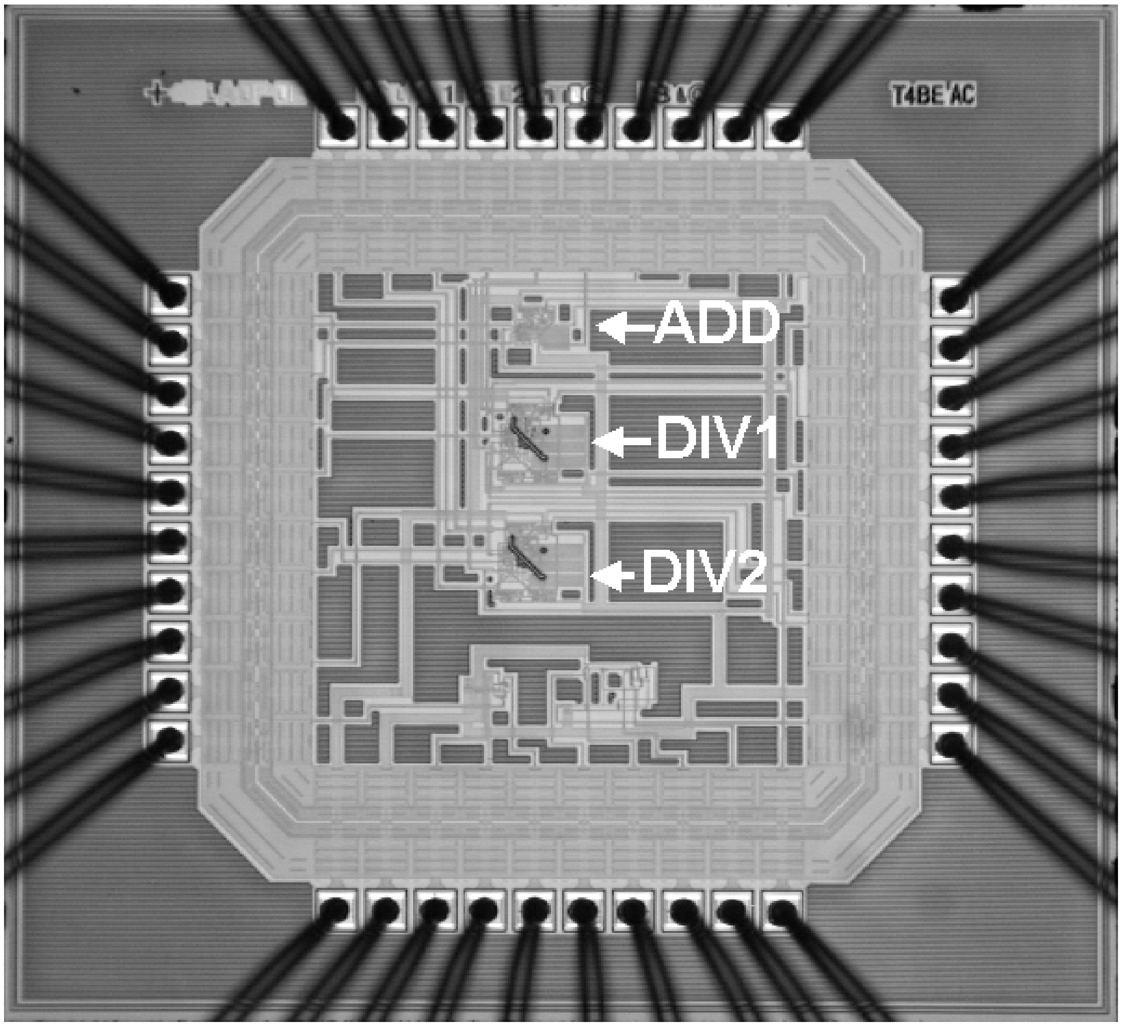


Figure 5.6. Chip microphotograph.

5.2. Simulation and Test Results

For error and accuracy results, we define the percentage error as the absolute error in the output voltage divided by the output full range and multiplied by 100%. The full range is equal to *$2V_M$* since the output changes from $-V_M$ to $+V_M$. Using (4.2) and (4.3), we can find that error and accuracy translate into time as

$$Error = \frac{\left|\left(t_{cal+} - t_{cal-}\right) - \left(t_{meas+} - t_{meas-}\right)\right|}{2T_f} \times 100\% \qquad (5.1)$$

$$Accuracy = \left(1 - \frac{\left|\left(t_{cal+} - t_{cal-}\right) - \left(t_{meas+} - t_{meas-}\right)\right|}{2T_f}\right) \times 100\% \qquad (5.2)$$

where t_{cal} is the calculated time and t_{meas} is the measured time. All the results presented here are for 3.2 V supply voltage.

5.2.1. IPI-to-Voltage and Voltage-to-IPI Conversions

Adding two inputs requires IPI-to-V conversion of each input in the first frame and V-IPI conversion of the voltage on the shared capacitor in the second frame. Figure 5.7 shows the operation of IPI-to-V-to-IPI conversion as addition of two equal inputs. In this case, t_{1+}= t_{2+}=60 ns, t_{1-}= t_{2-}=40 ns, and the output t_{3+}=71 ns. Accuracy in this case is 98%. Figure 5.8 is a graph of the chip test and simulation results for addition when the first input is varied while the second input is fixed at zero (t_{2+}= t_{2-}= 50 ns). Worst case accuracy from test is better than 96%, and from simulation, it is better than 98%. This miscorrelation between test and simulation is due to capacitive loading effects from the package pins, the wires, and the oscilloscope. Test and simulation results are optimized to have optimum accuracy around the zero point by adjusting V_1 to 980 mV and 960 mV, respectively. This also applies to the addition and subtraction results in the next subsection. The IPI-to-V conversion part of the circuit occupies 4.544x10^{-3} mm^2 of chip area and consumes 0.96 mW of power. The V-to-IPI conversion part of the circuit occupies 5.12x10^{-3} mm^2 of chip area and consumes 2.24 mW of power. Each conversion operation takes only one time frame to finish. The dynamic range of both conversions is 1200 mV.

5.2.2. Addition and Subtraction

As described above, Figure 5.7 shows the addition of two equal inputs and Figure 5.8 is a graph of the chip test and simulation results for this operation along with the calculated ideal results. Worst case accuracy from test is better than 96%, and from simulation, it is better than 98%. Subtraction was tested using the addition circuit by simply inverting the second input. Figure 5.9 shows the subtraction of two equal inputs. In this case, $t_{1+}= t_{2+}=78$ ns, $t_{1-}= t_{2-}=22$ ns, and the output $t_{3+}=49$ ns. Accuracy in this case is 98%. Figure 5.10 is a graph of the chip test and simulation results along with the calculated results for subtraction when the first input is set to zero ($t_{1+}= t_{1-}= 50$ ns) while the second input is varied. Worst case accuracy from test is better than 96%, and from simulation, it is better than 98%. The addition/subtraction circuit occupies $7.68\text{x}10^{-3}$ mm^2 of chip area and consumes 1.92 mW during the first frame and 2.24 mW during the second frame. So, the average power consumption is 2.08 mW. Addition/subtraction takes two time frames to finish. The dynamic range of operation is 1200 mV.

5.2.3. Division and Multiplication

Figure 5.11 shows the division operation. In this case, $t_{1+}= t_{2+}=92$ ns, $t_{1-}= t_{2-}=8$ ns, $K_{div1}=33.3$ ns and the output $t_{3+}=88$ ns. The accuracy of division in this case is 95.3%. Figure 5.12 is a graph of the chip test and simulation results for division along with the calculated ideal results when the first input is varied while the second input is fixed ($t_{2+}=$ 69ns). Worst case accuracy from test is better than 96%, and from simulation, it is better than 98%. Figure 5.13 is a graph of the chip test and simulation results for multiplication along with the calculated ideal results when the two inputs are equal and T_{const+} is fixed at 69ns. Worst case accuracy from test is better than 95%, and from simulation, it is better than 97%. The division circuit occupies $25.92\text{x}10^{-3}$ mm^2 of chip area and consumes 2.88 mW during the first frame and 3.84 mW during the second frame. So, the average power consumption is 3.36 mW. Multiplication occupies twice the area and consumes twice the power. It occupies 51.84x10^{-3} mm^2 of chip area and consumes 6.72 mW of power. Division and multiplication each take two time frames to finish. Their dynamic range is 600 mV.

5.2.4. Summary

We have demonstrated the functionality and implementation of the IPI based conversion and computation methods and circuits presented in chapter 4 by fabricating them in a chip using the TSMC 0.35 um mixed-signal CMOS process and by testing the chip at 10 MHz framing speed, which corresponds to 100 ns time frame. The test results agree well with both theoretical results and SPICE simulation results. However, accuracy from test is 2% worse than predicted by simulation. We believe that this miscorrelation between test and simulation is due to capacitive loading effects from the package pins, the wires, and the oscilloscope. Table 5.1 is a summary of the accuracy, power consumption, area, dynamic range, and speed results for all circuits.

0.35 um CMOS process V_{DD} = 3.2 V	Accuracy (%)	Average power (mW)	Area (10^{-3} mm^2)	Speed (MHz) (Mop/s)	Dynamic range (mV)
IPI-to-V	*98*, 96	*0.96*	*4.544*	*100*, 10 *100*, 10	1200
V-to-IPI	*98*, 96	*2.24*	*5.12*	*100*, 10 *100*, 10	1200
Addition/Subtraction	*98*, 96	*2.08*	*7.68*	*100*, 10 *50*, 5	1200
Division	*98*, 96	*3.36*	*25.92*	*50*, 10 *25*, 5	600
Multiplication	*97*, 95	*6.72*	*51.84*	*50*, 10 *25*, 5	600

Table 5.1. Summary of accuracy, power consumption, area, dynamic range, and speed results of the IPI circuits. Results in *Italic* are from simulation. 4-Q division and multiplication need about 60% more area and consume about 45% more power.

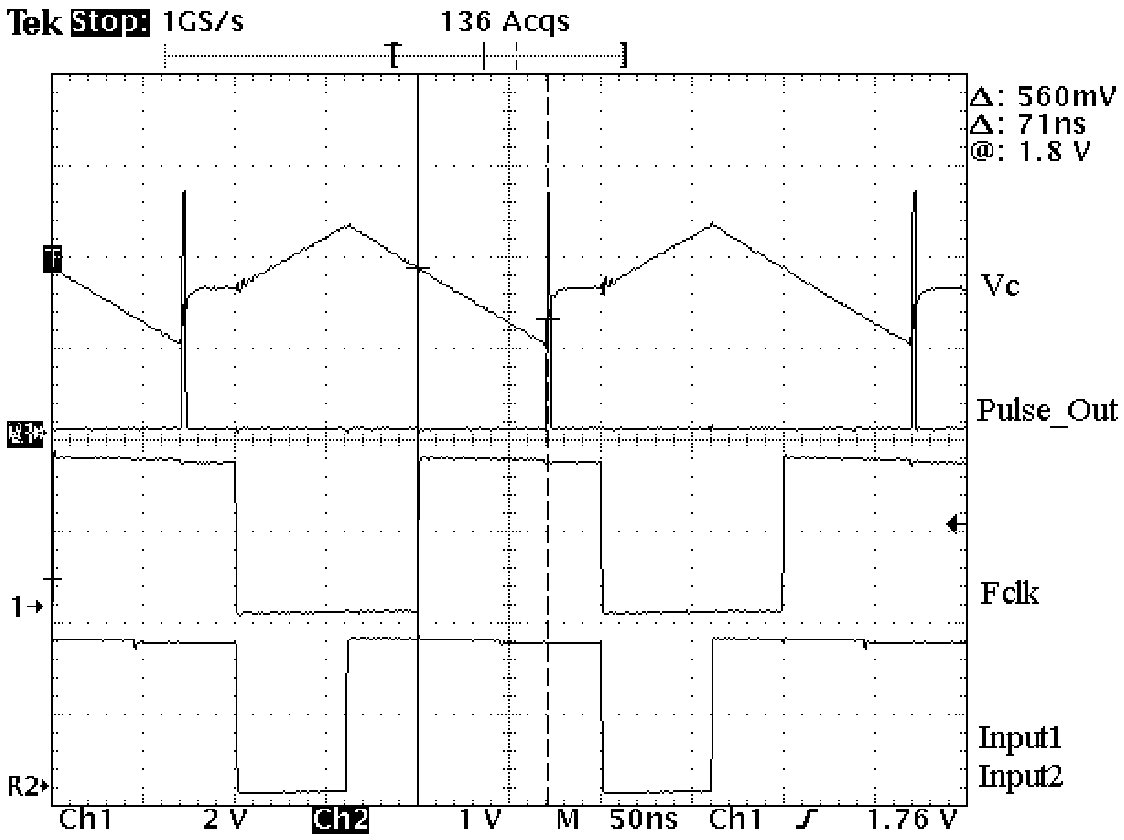


Figure 5.7. Oscilloscope screen image showing the addition of two equal inputs. IPI-to-V conversion of the two inputs is done in the first frame and V-to-IPI conversion of the output voltage in the second.

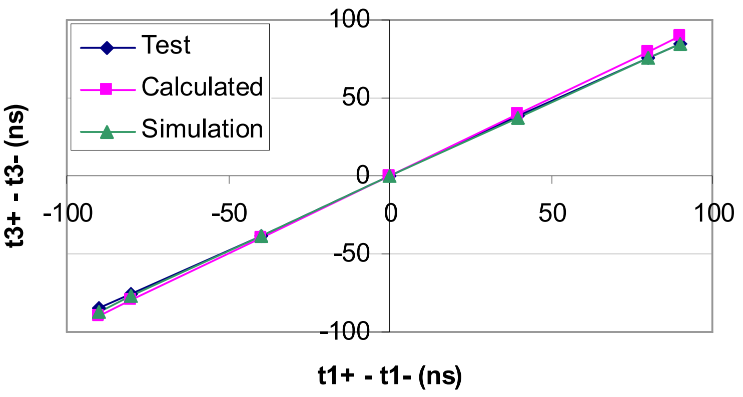


Figure 5.8. Test, simulation, and calculated results for addition when the first input is varied while the second input is fixed at zero.

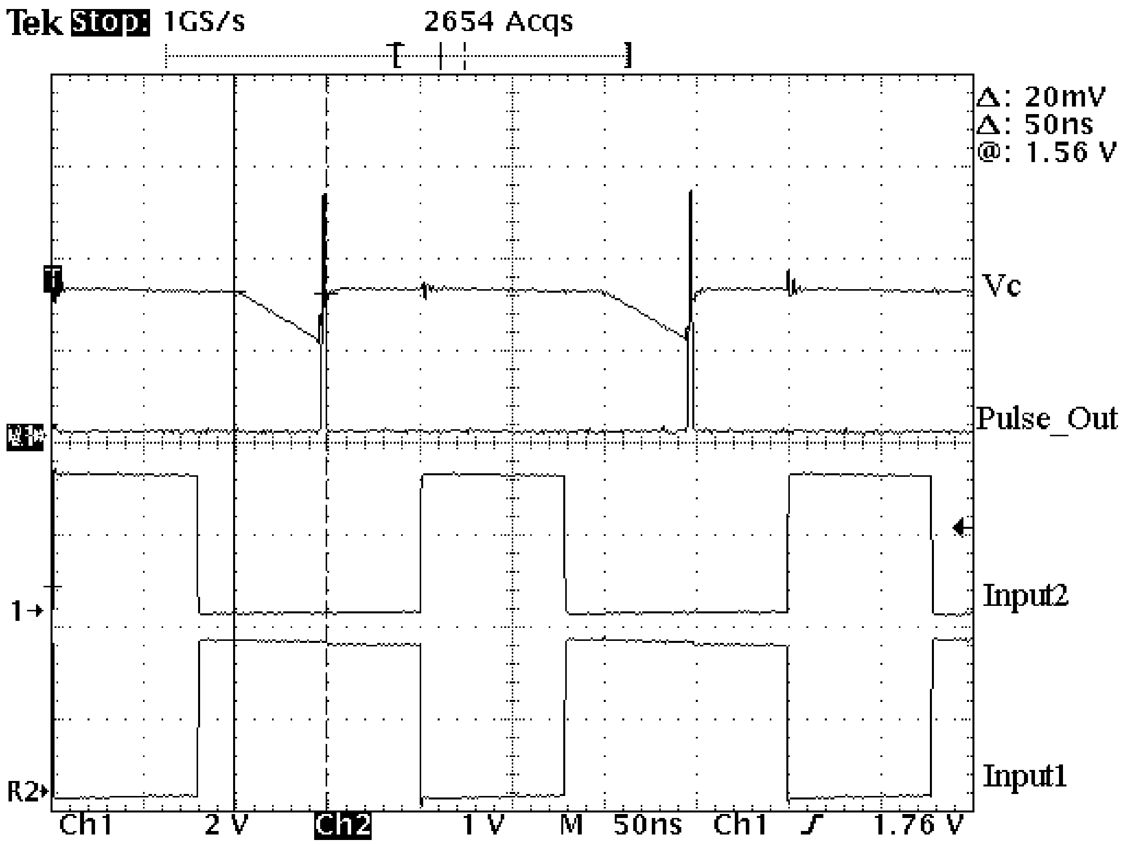


Figure 5.9. Oscilloscope screen image showing the subtraction of two equal inputs. The output voltage remains flat at V_{middle} during the first frame since the two inputs are equal and therefore canceling each other.

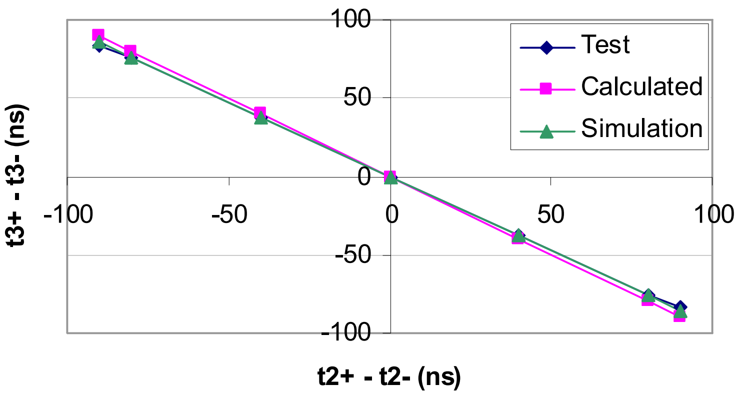


Figure 5.10. Test, simulation, and calculated results for subtraction when the second input is varied while the first input is fixed at zero.

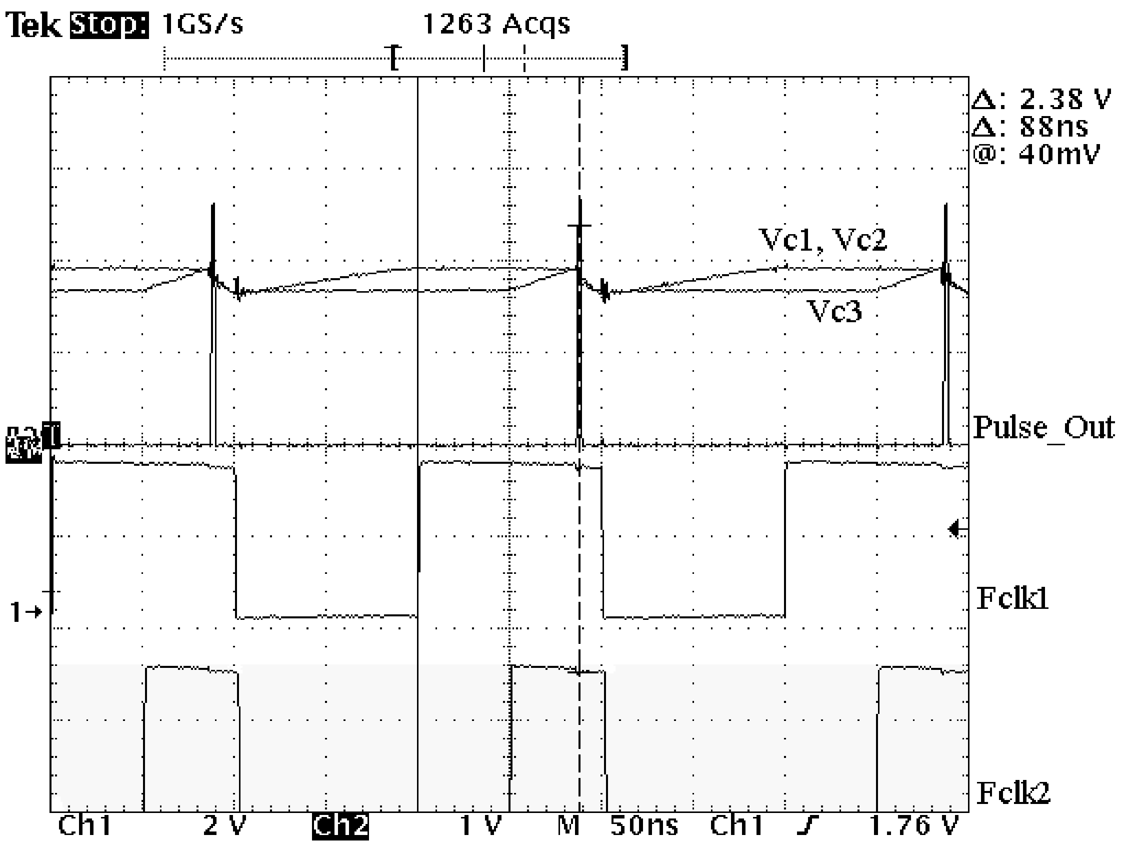


Figure 5.11. Oscilloscope screen image showing the division of two equal inputs.

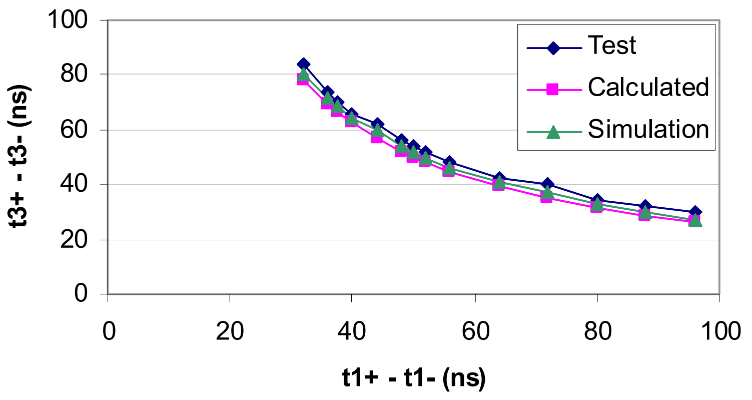


Figure 5.12. Test, simulation, and calculated results for division when the first input is varied while the second input is fixed (t_{2+}= 69ns).

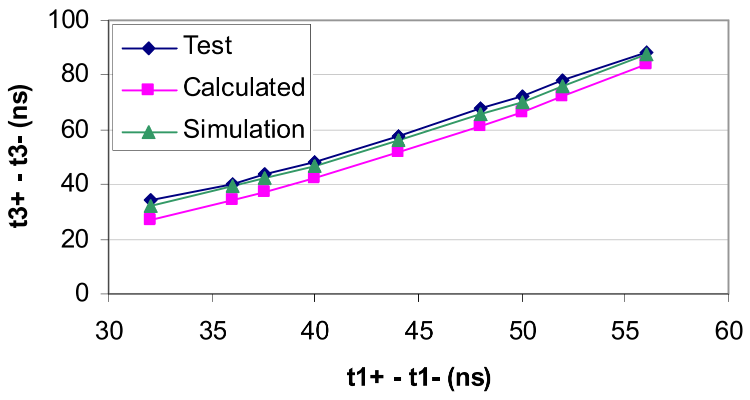


Figure 5.13. Test, simulation, and calculated results for multiplication when the two inputs are equal and T_{const+} is fixed at 69ns.

6. System-Level Design and Applications

In this Chapter, we will describe a system-level design which executes a computation based on an artificial neuron with multiple synapses. The design incorporates the basic arithmetic building blocks we have developed in chapter 4 and integrates them together into a simple system. We will also provide the SPICE simulation results based on the same TSMC 0.35um CMOS technology that was used for the basic IPI building blocks. We will also discuss how the IPI technology can be used in applications such as sensors, instrumentation, communications, telemetry, signal processing, and ANNs. To demonstrate the advantages of using the IPI technology in these applications, we will compare our IPI based conversion and computation implementations against other analog and digital implementations.

6.1. System-Level Design

6.1.1. Artificial Neural Networks (ANNs)

ANNs get their computational processing power from the fact that they try to resemble real biological neural networks found in the brain. They have been successfully used in many real life applications such as data classification, pattern recognition, function approximation, and adaptive signal processing [4] and [5]. An ANN is built from several layers of neurons each connecting to each by synapses. Figure 6.1 shows an example of ANN architecture (topology). This type is the most popular and is called feed-forward or multi-layer perceptron (MLP) network.

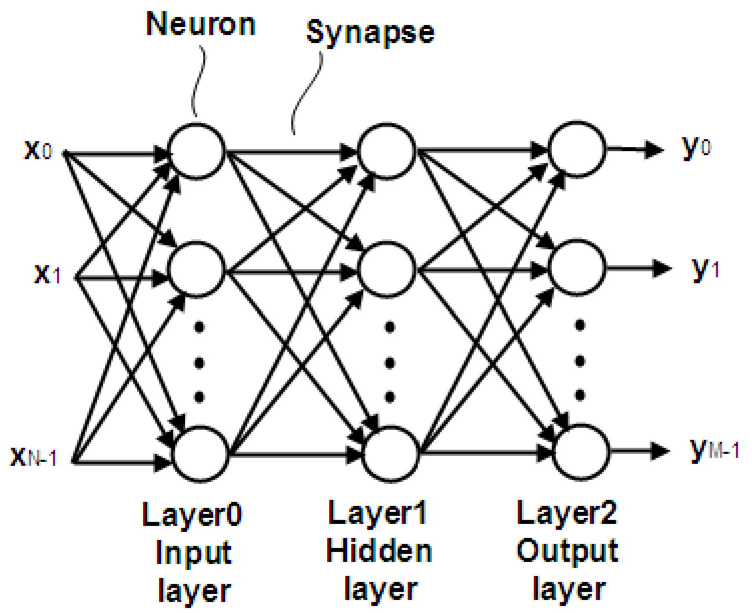


Figure 6.1. Feed-forward or multi-layer perceptron (MLP) ANN.

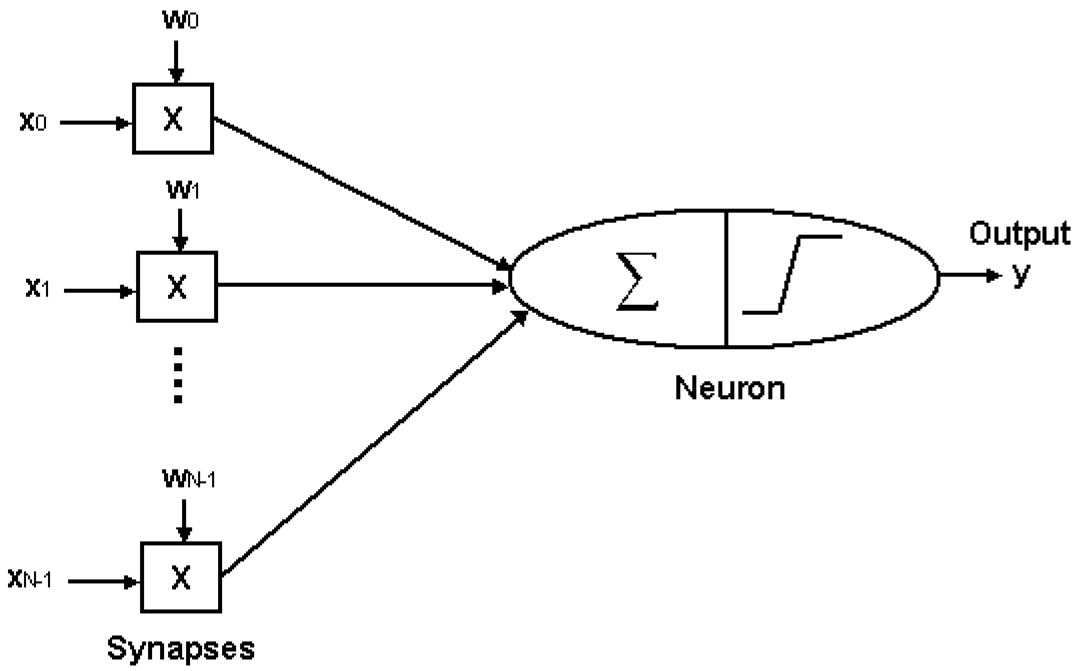


Figure 6.2. Computational model of an abstract neuron and its synapses.

Figure 6.2 is a computational model of the neuron and its synapses. It has N inputs ($x_0, x_1, \ldots, x_{N-1}$) . To model the synaptic strength of the connections, each input x_i is weighted (multiplied) by a weight w_i. Then, the neuron combines (sums) the weighted inputs and applies a nonlinear transformation function f on the summation result to generate the output y. In mathematical terms, the neuron can be modeled using the following equation

$$y = f\left(\sum_{i=1}^{N-1} w_i \cdot x_i\right) \tag{6.1}$$

Figures 6.3 and 6.4 show two of the most popular neuron transformation functions, the sigmoid function defined as

$$f(sum) = \frac{1}{1+e^{-sum}} \tag{6.2}$$

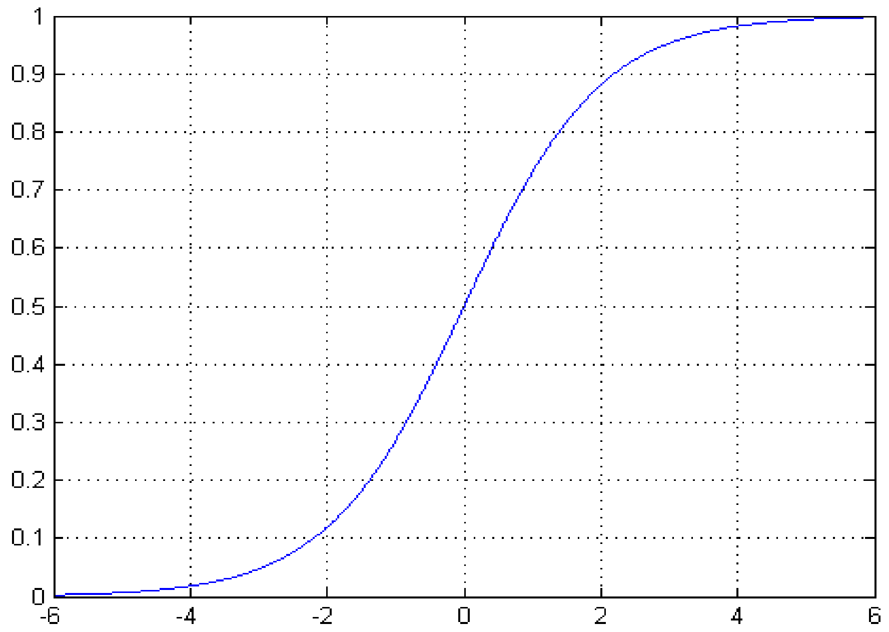


Figure 6.3. The sigmoid function.

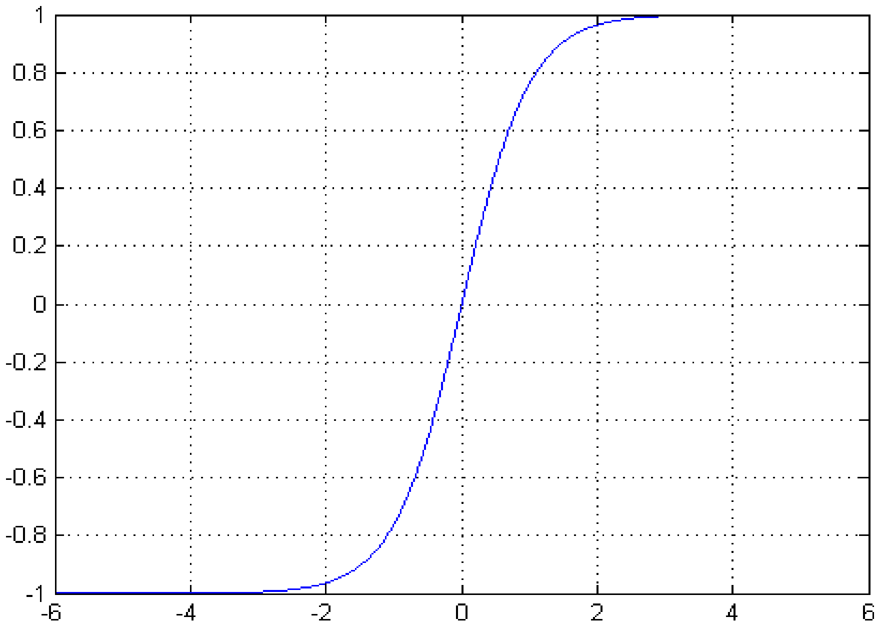


Figure 6.4. The hyperbolic tangent function.

and the hyperbolic tangent function defined as

$$f(sum) = \tanh(sum) = \frac{e^{-sum} - e^{-sum}}{e^{-sum} + e^{-sum}} \qquad (6.3)$$

The following characterizes the neuron transformation function: It is nonlinear, it is monotonically increasing for the range of operation, and it saturates to a minimum or a maximum when the sum magnitude becomes large in the negative or positive direction respectively.

In pattern recognition and classification applications, the inputs are the features of the object to be classified. Each output neuron is associated with a class. The neuron in the output layer with the strongest output indicates the class of the object. In function approximation, the inputs are the input variables of the function to be approximated and the output of the neuron is the output of the function to be approximated. Before an ANN reaches to the optimum solution (optimum weight values assuming the topology is fixed), it needs to learn (we need to train it). This can be done by providing it with the desired output *d* and comparing its output *y* with it. The error (difference between *y* and *d*) can then be used to adjust the synaptic weights until the error is minimal. One way of doing this is the error back-propagation algorithm proposed by Rumelhart et al. in [52]. Such learning algorithms are beyond the scope of this dissertation and are not discussed further. Our objective here is to demonstrate the operation of the basic IPI building blocks as a system.

6.1.2. System-Level Design and Simulation

In this section, we will put the basic IPI building blocks together in a design that can perform the computational functions of the neuron and its synapses as defined in (6.1) and shown in Figure 6.2. In this design example, the neuron has six inputs (the number of synaptic connections *N* is equal to 6). Figure 6.5 shows the design, Figure 6.6 shows its simulation, and Figure 6.7 is a timing diagram that shows the phases and sequence of the computations. Six IPI multiplication circuit instances are used to model the strength of the six synaptic connections. The input x_i and the synaptic weight w_i are both in IPI form. The outputs of all the multiplications are fed into a six-input IPI addition circuit. The addition circuit performs both functions of the neuron, summation and transformation. We were able to exploit the nonlinearity and saturation properties of the circuit to achieve the desired nonlinear and saturating behavior needed in the transformation function of the neuron.

The system runs at the speed of one operation per two frames. It is only the first result that takes four frames to come out but after that results come out every two frames. This is because the two division blocks (in each multiplier) and the addition block operate in a pipelined fashion. As a block is producing its result and providing it to the receiving block in one frame, the receiving block is taking this result as input and performing the IPI-to-V conversion of the inputs in the same frame. This pipelining can be seen by looking at the timing diagram in Figure 6.7, and is analogous to a pipelined digital processor in the pipe latency is *N* clock cycles, where *N* is the number of stages [121]. Once the pipe is full, one instruction finishes every clock cycle.

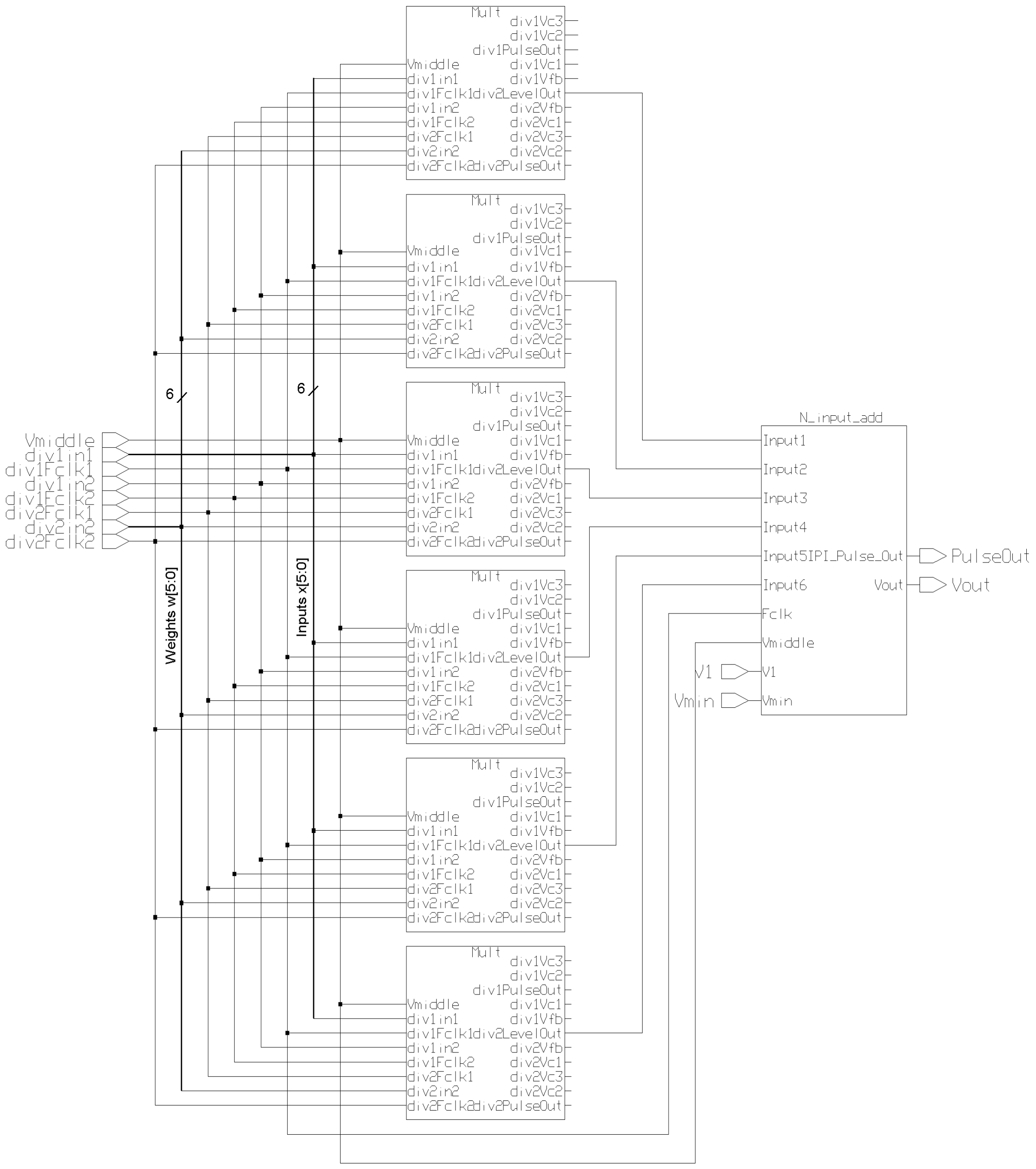


Figure 6.5. System-level design of one neuron with 6 synapses using the IPI addition and multiplication circuits.

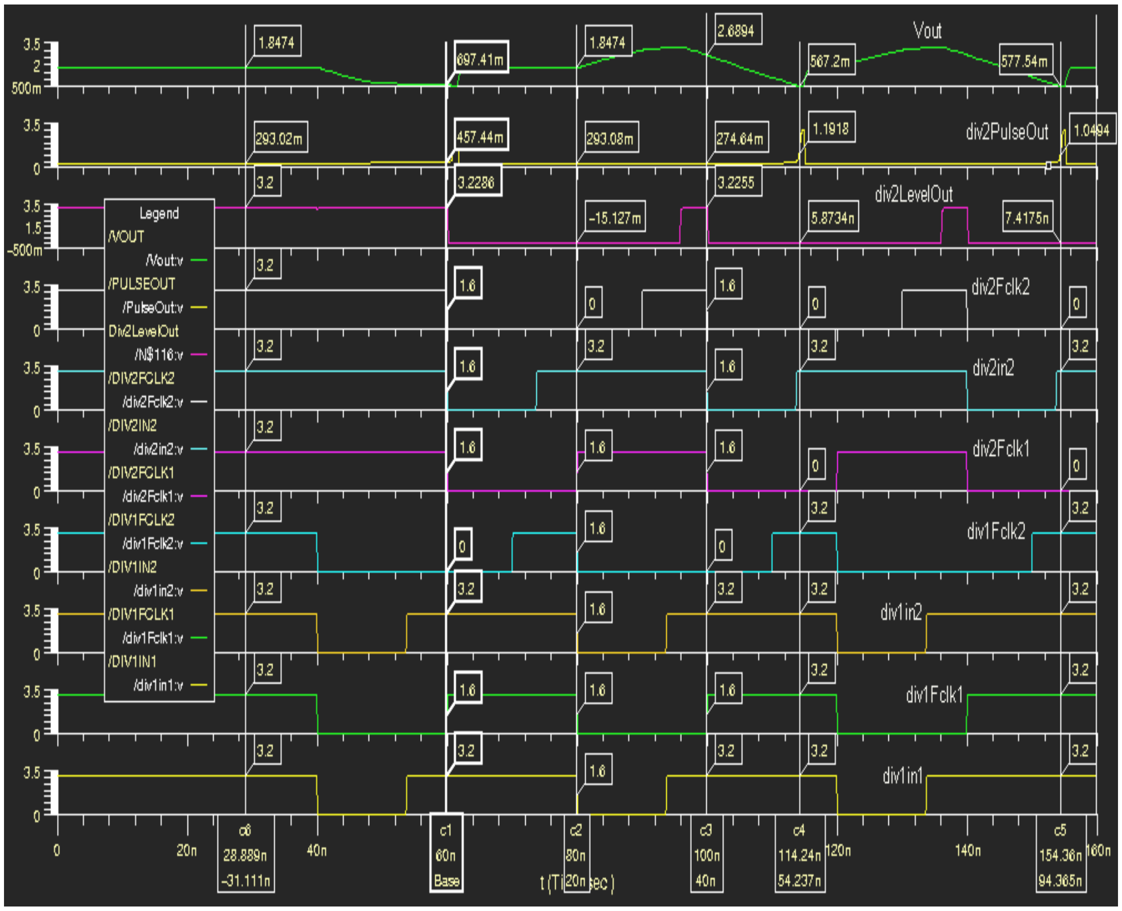
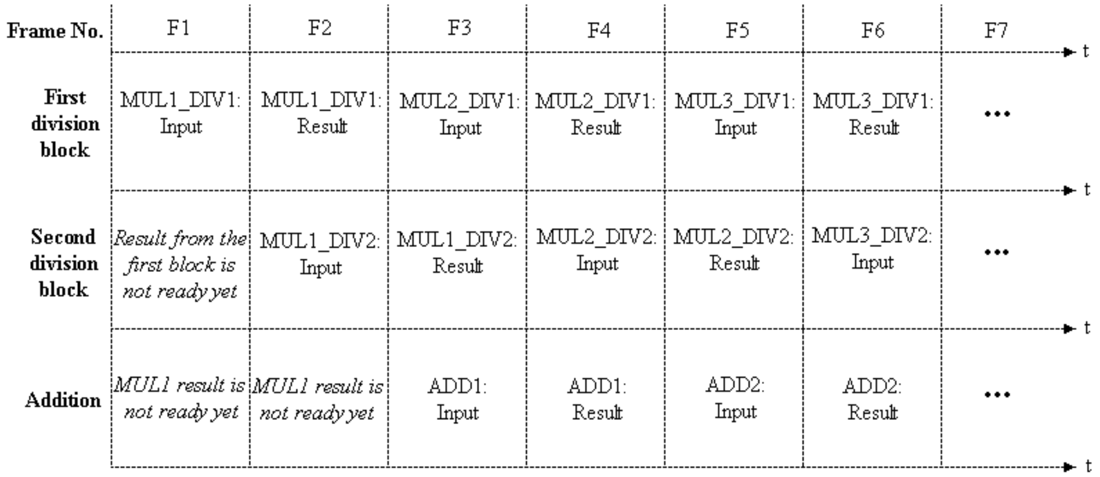


Figure 6.6. Simulation of the system-level design of one neuron with 6 synapses.



Frame No.	F1	F2	F3	F4	F5	F6	F7
First division block	MUL1_DIV1: Input	MUL1_DIV1: Result	MUL2_DIV1: Input	MUL2_DIV1: Result	MUL3_DIV1: Input	MUL3_DIV1: Result	...
Second division block	*Result from the first block is not ready yet*	MUL1_DIV2: Input	MUL1_DIV2: Result	MUL2_DIV2: Input	MUL2_DIV2: Result	MUL3_DIV2: Input	...
Addition	*MUL1 result is not ready yet*	*MUL1 result is not ready yet*	ADD1: Input	ADD1: Result	ADD2: Input	ADD2: Result	...

Figure 6.7. System timing diagram.

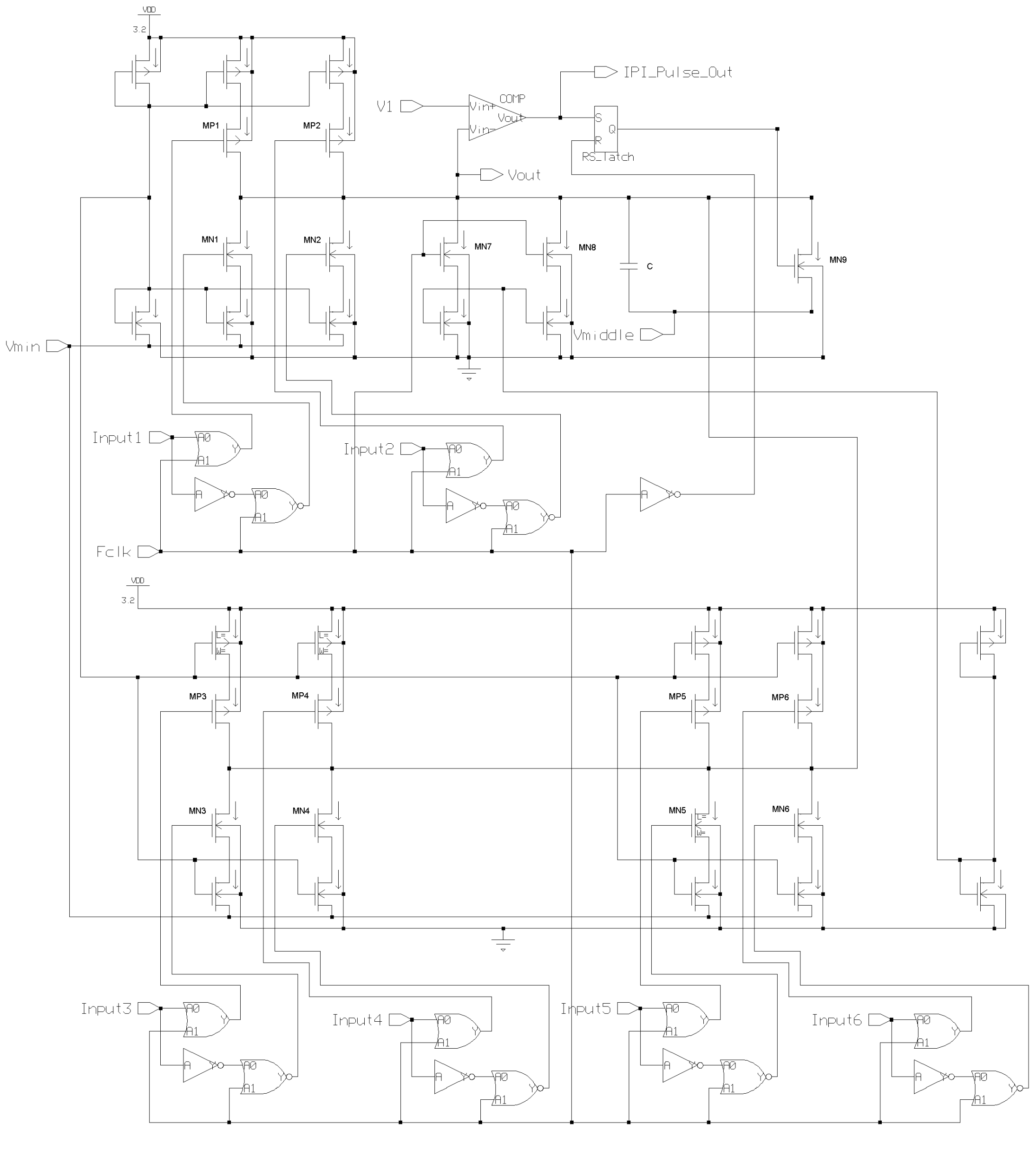


Figure 6.8. 6-input IPI addition which performs the summation and transformation functions of a 6-synapse neuron.

Figure 6.8 shows the 6-input addition (6-synapse neuron) circuit which is based on the 2-input addition circuit presented in section 4.2. Figures 6.9, 6.10, and 6.11 show three simulation cases: i) the summation result saturates to the minimum when all inputs are large and negative, ii) the summation result is in the middle of the range when all inputs are zero, and iii) the summation result saturates to the maximum when all inputs are large and positive. Figures 6.12 and 6.13 are graphs of the simulation results for the voltage and the pulse time output over the full dynamic range. The graphs show a behavior similar to that of the sigmoid and hyperbolic tangent transformation functions shown in the previous subsection. One difference though which can be easily seen by looking at the simulation results in Figure 6.13, is that the output is not zero when all inputs are zeros. It is negative. This is due to some nonlinearity in the output voltage of the addition circuit which happens during the first half of the input frame but does not happen in the second half. During the first half of the input frame, all inputs cause the output capacitor to be charged in the positive direction. The voltage on the capacitor is linear initially, but then becomes nonlinear when it is high enough to reduce the source-drain voltage of the PMOS current sources and cause them to operate in the triode region. This decreases the positive current charging the capacitor. On the other hand, during the second half of the input frame, all inputs cause the capacitor to be charged in the negative direction. The voltage on the capacitor stays linear because the source-drain voltage of the NMOS current sources remains high enough to keep them in the saturation region. This keeps the negative current charging the capacitor constant. The difference in the output is negative because the negative current remains constant while the positive current decreases when the PMOS current sources begin to operate in the triode region.

This inaccuracy will be a problem for ANNs which use back-propagation based learning [52] but not for those which use weight perturbation based learning [60-63], since the back-propagation method requires the derivative of the neuron transfer function to compute the gradient descent of the error with respect to the weight, while the weight perturbation method approximates the gradient descent using the finite differences. A perturbation is injected at the weight and the error in the network output is measured before and after the injection to approximate the gradient descent [4].

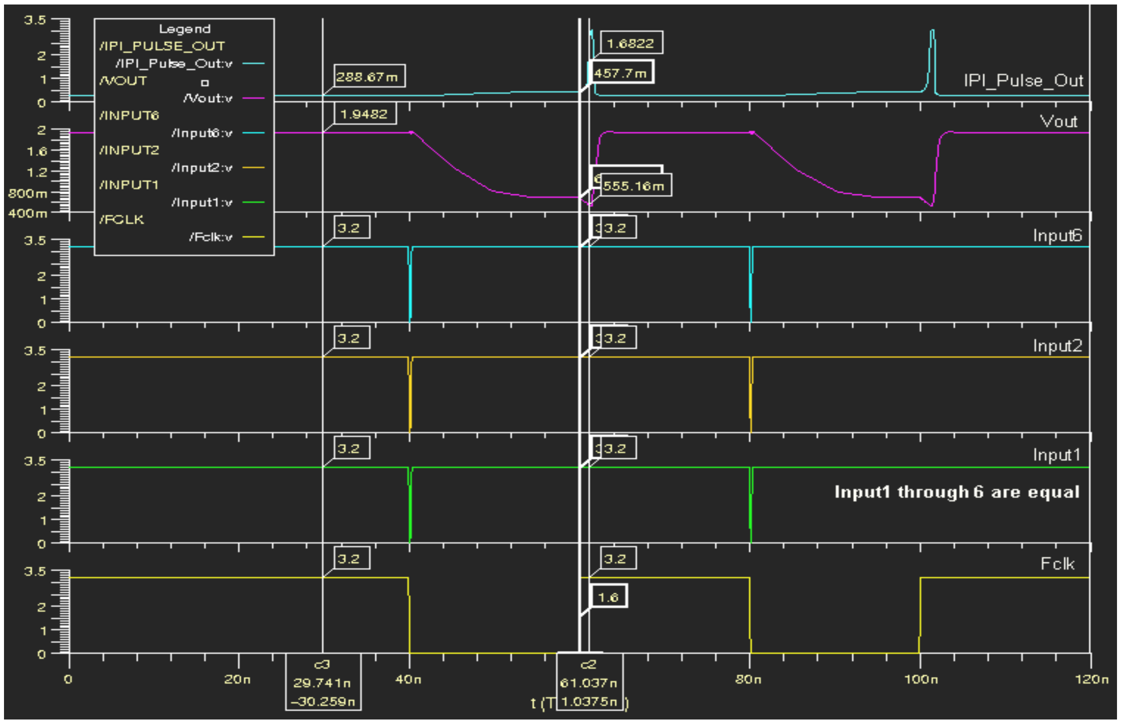


Figure 6.9. Simulation of the 6-synapse neuron: summation result saturates down to the minimum when all inputs are large in negative.

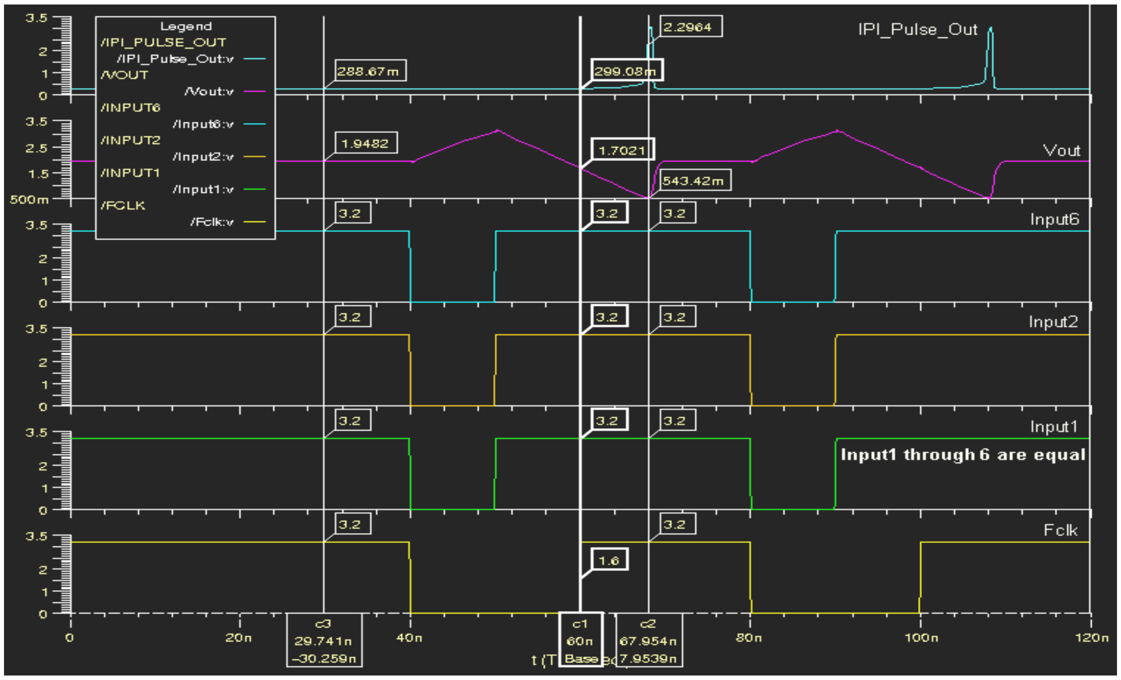


Figure 6.10. Simulation of the 6-synapse neuron: summation result is in the middle of the range when all inputs are zero.

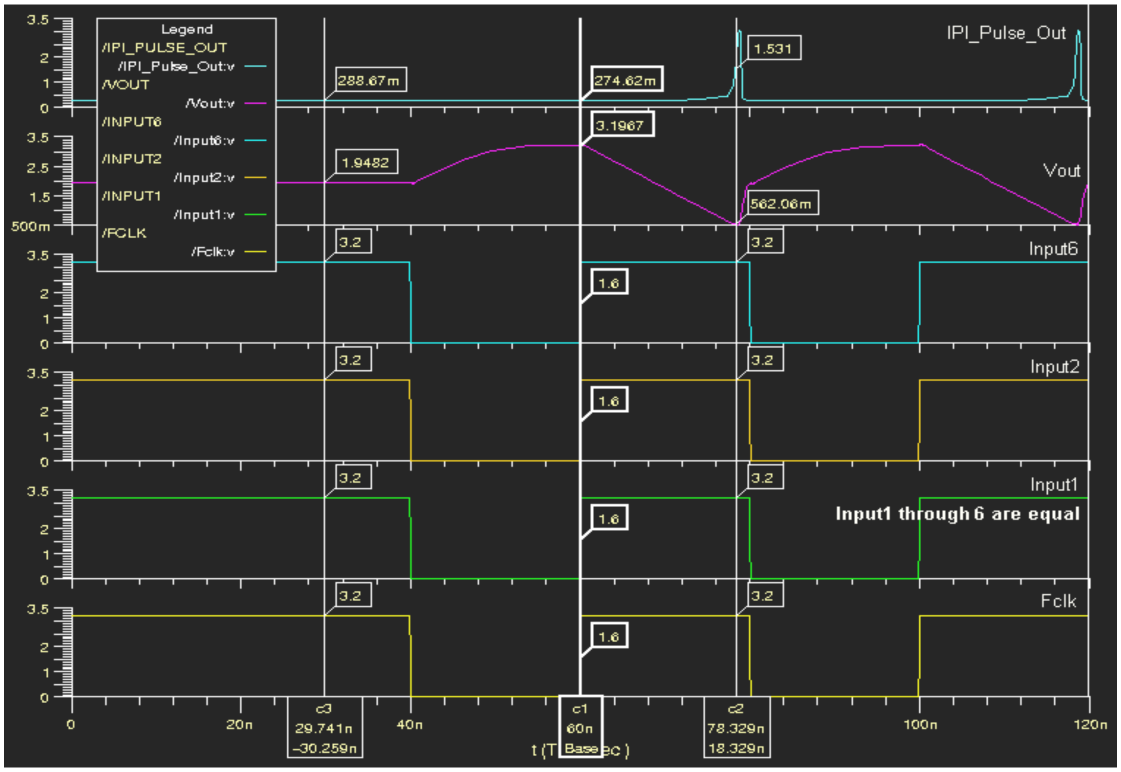


Figure 6.11. Simulation of the 6-synapse neuron: summation result saturates up to the maximum when all inputs are large in positive.

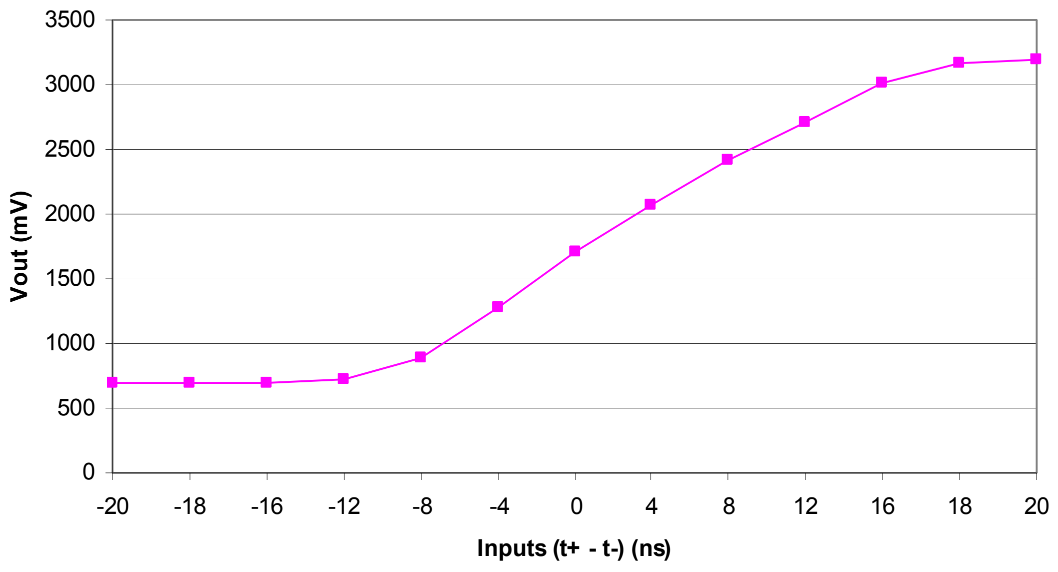


Figure 6.12. Simulation results of the 6-synapse neuron: Vout versus inputs $(t_+ - t_-)$.

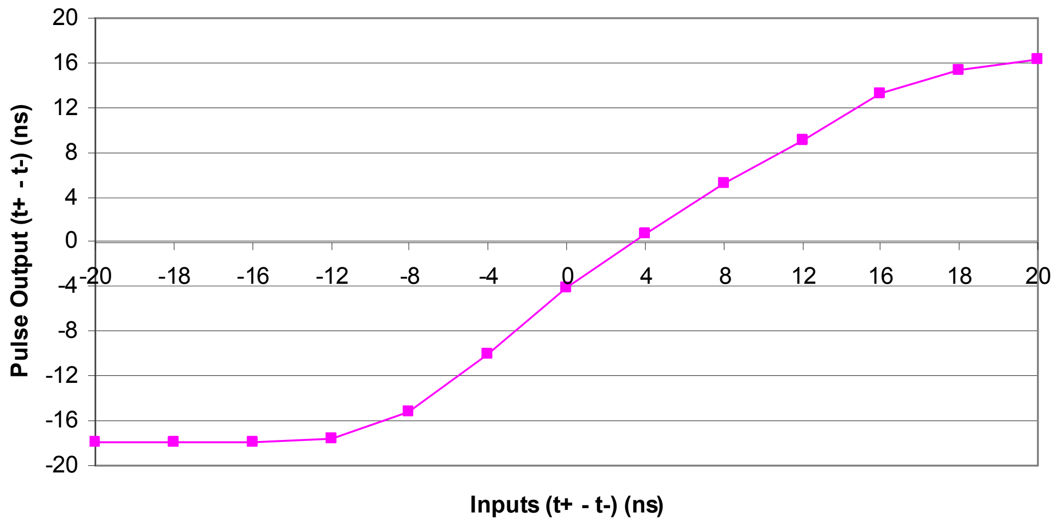


Figure 6.13. Simulation results of the 6-synapse neuron: pulse output $(t_+ - t_-)$ versus inputs $(t_+ - t_-)$.

6.1.3. Training and Programming and their Impact on Storage and Resolution

The way we train and program an ANN has a big impact on the cost and complexity of memory storage and therefore on resolution. ANNs can learn or be trained to perform a certain task by providing it with a desired output. The learning algorithm such as the error back-propagation in [52] can be used to calculate the new values of synaptic weights based on the error. This process of updating the synaptic weight values repeat until the error reaches to some acceptable minimum value. Assuming that the topology of the network is fixed, programming the network is then done by the synaptic weights. There are two types of programmable ANNs: non-adaptive and adaptive. Non-adaptive networks learn once and then they "freeze" the optimum weights. Adaptive networks learn continuously, updating the synaptic weights during normal operation. Many applications require this kind of adaptive signal processing as in [53] and [54] for example.

Non-adaptive networks require memory cells that can store the programming information (the synaptic weights) for an indefinite period of time. Long term memory cells can be digital such as dynamic or static random access memory (RAM) or they can be analog such as self-refreshing capacitor based cells as in [55] and [56], which rely on the periodic refresh of the voltage on a capacitor to predefined quantized voltage levels to avoid loss of voltage (charge) through leakage currents. The resolution is defined as

$$resolution = \frac{\delta}{FR} \tag{6.4}$$

where δ is the step size and FR is the full range. Therefore, for networks that use binary values for the synaptic weights, the resolution is

$$resolution = \frac{1}{2^{-N}} \tag{6.5}$$

where N here is the number of bits. For example, if $N=6$ then the resolution is 1.56%. For networks that use self-refreshing capacitor based analog cells, the resolution depends on the steps between the predefined quantized voltage levels. If the maximum step size is 20 mV and the full range is 1 V then the resolution is 2%.

Although we are not implementing the learning algorithm hardware here, we suggest that such an implementation should generate the new values for synaptic weights in the IPI form. This will allow weights to be stored in the capacitor that corresponds to the weight input in the IPI multiplication circuit so that no extra capacitors are needed for storing the weights since the capacitor is already a part of the multiplication circuit.

6.1.4. Estimation of Accuracy (Resolution), Power, Area, Dynamic Range, and Speed

As we explained above, the overall accuracy (or resolution) of the final outputs (solutions) of the full ANN will depend mainly on the resolution of the synaptic weight signals generated by the learning hardware and how the output of the neuron changes with respect to their change. In IPI form, the full range of the weight signal in our system-level simulation is 20ns which corresponds to 600mV. For non-adaptive networks which require long-term storage of the weight value, the capacitor voltage need to be refreshed to one of the pre-defined quantized voltage levels. If we choose the number of levels to be 64 for example then the step size would be about 9mV and the resolution would be 1.56%. For adaptive networks, no refreshing is needed and the resolution should be better since the weight value is not quantized, i.e., it has a continuous range of values. However, the learning hardware will be more complex, since it needs to learn and keep updating the weights while the network is working.

The system-level design consumes 6.72 mW per synapse (multiplier) and 1.76 mW per neuron (adder) plus 0.16 mW per adder input. So, the total average power consumption of the system-level design of one neuron with 6 synapses is 43 mW. In general, the following formula can be used to compute the total average power consumption in mW for a system of one neuron with *N* synapses

$$P_{Total} = 6.72N + (1.76 + 0.16N) \qquad (6.6)$$

The power consumed per synapse is about 4 times the power consumed by the neuron.

The system-level design occupies 51.84x10^{-3} mm^2 per synapse and 6.58x10^{-3} mm^2 per neuron plus 0.55x10^{-3} mm^2 per neuron input. So, the total area needed for the system of one neuron with 6 synapses is 321x10^{-3} mm^2. In general, the following formula can be used to compute the total area in 10^{-3} mm^2 for a system of one neuron with N synapses

$$A_{Total} = 51.84N + (6.58 + 0.55N) \tag{6.7}$$

The area needed per synapse is about 8 times the area of the neuron.

From the simulation results graph in Figure 6.11, the dynamic range of the output including the nonlinear and saturation regions is 2.5 V. Table 6.1 has a summary of the system-level resolution (based on the assumption of 64 quantized capacitor refresh levels), power consumption, area, dynamic range, and speed results for one neuron with 6 synapses.

Resolution (%)	Average Power (mW)	Area (10^{-3} mm^2)	Frame speed (MHz)	Dynamic range (V)
1.56	43	321	50	2.5

Table 6.1. Summary of system-level resolution, power consumption, area, dynamic range, and speed results for one neuron with 6 synapses.

6.2. Applications and Comparisons with Other Implementations

The IPI technology can be useful in many applications. Among these applications are sensors, instrumentation, communications, telemetry, signal processing, and ANNs. In communication, the signal can be sampled and converted to an IPI pulse stream using the V-to-IPI converter (modulator). The pulse transmission method can be electrical (wired or wireless), acoustic, or even optical if necessary. Along the way, the pulses can be refreshed using a digital buffer. On the receiver side, the SNR can be improved by averaging the pulse stream. If necessary, the pulse stream can be converted back to the analog domain using the IPI-to-V converter (demodulator). In sensors/instrumentation, the sensor senses the analog quantity (temperature, pressure, light, etc) and generates a voltage or current quantity. If it is current then it can be easily converted to voltage by directing it through a resistor or by accumulating it as charge on a capacitor. The voltage signal is then sampled and converted to an IPI pulse stream using the V-to-IPI converter. Telemetry combines both instrumentation and communications, and can be performed as we just explained for both of them. If the distance is very long and inserting repeaters (digital buffers) along the pulse trip is not possible then optical communication can even be used and the pulses can be transmitted as light pulses (infra-red or laser). In signal processing, the dot-product and the multiply-and-accumulate (MAAC) operations are heavily used operations. The IPI addition and multiplication can be used together to perform these operations in the time domain, as we demonstrated for ANNs in the previous section.

In this section, we will compare our IPI based conversion and computation implementations with other analog and digital implementations. We will also tie our comparison findings with the applications that we just discussed above. For detailed descriptions of the analog and digital implementations with which we are comparing, the interested reader is referred to the reference section. Since addition and subtraction are easier to design and less expensive in terms of power, area, and speed than multiplication and division, we will focus our comparison on multiplication and division. When comparing against digital implementations, we will also compare V-to-IPI against analog-to-digital conversion (ADC).

6.2.1. Comparison with Analog Implementations

Mead in [5] and Jabri et al. in [4] describe several analog techniques and circuits for arithmetic computation including addition and multiplication. Addition and subtraction of voltages and currents in the analog domain can be performed by simply utilizing Kirchoff's voltage and current laws, respectively. As we will see in this subsection, analog computation in CMOS circuits depends directly on the MOSFET transistor I-V characteristics in order to implement a certain arithmetic operation. The following are simple equations that approximately model the MOSFET transistor I-V characteristics in the triode and the saturation regions, respectively,

$$I = \mu C_{ox} \frac{W}{L}\left[(V_{GS} - V_t)V_{DS} - \frac{V_{DS}^2}{2}\right] \qquad (6.8)$$

$$I = \mu C_{ox} \frac{W}{L}(V_{GS} - V_t)^2 \qquad (6.9)$$

where V_{GS} is the gate to source voltage, V_{DS} is the drain to source voltage, V_t is the threshold voltage, W is the channel width, L is the channel length, μ is the carrier mobility, and C_{ox} is the gate oxide capacitance per unit area and is equal to

$$C_{ox} = \frac{\varepsilon_{ox}}{t_{ox}} \qquad (6.10)$$

where ε_{ox} and t_{ox} are the oxide permittivity and thickness, respectively.

Han and Sanchez-Sinencio in [26] survey all types and architectures of analog CMOS multipliers. All analog multipliers use the same basic idea. Similar to the transconductance multiplier that we described in subsection 2.2.2, all analog multipliers utilize the MOSFET transistor I-V characteristics in the triode or the saturation region, to produce intermediate results. Then, they use addition and subtraction to add and subtract these intermediate results to or from each other, to cancel undesirable nonlinear terms and keep only the desirable linear term kxy, where k is a scaling factor and x and y are the two analog inputs to be multiplied. The following equation is an example of this,

$$z = k\begin{bmatrix} \left([(X+x)+(Y+y)]^2 + [(X-x)+(Y-y)]^2\right) \\ -\left([(X-x)+(Y+y)]^2 + [(X+x)+(Y-y)]^2\right) \end{bmatrix} = 8kxy \qquad (6.11)$$

where X and Y are DC common mode signals, and x and y are the small signal inputs to be multiplied. The DC common mode signals are also used to correctly bias the transistors that will receive the inputs. Here is how (6.11) can be implemented using MOSFET devices:

1. Differential small signals $\pm x$ and $\pm y$ need to be generated and DC-shifted by X and Y, respectively.
2. Then, four voltage adders are used to calculate the four terms ($X \pm x + Y \pm y$).
3. Then, individual results from the adders are squared by applying them to the gates of identical MOSFET transistors operating in their saturation region. This will produce four currents according to (6.9).
4. Then, these currents (squaring results) are added and subtracted from each other according to equation (6.11). The output current will be $8kxy$ where the scaling factor k in this case is equal to

$$k = \mu C_{ox} \frac{W}{L} \tag{6.12}$$

Table 6.2 is a comparison of the IPI multiplier versus analog CMOS multipliers: [28] is based on the Gilbert multiplier cell [27], [29] is a transconductance multiplier operating in the linear (triode) region as does [14] and [15], [30] and [31] are transconductance multipliers operating in the saturation region, and the last three multipliers are described in [110] and [111] and they are transconductance multipliers operating in the saturation region and based on the Vgs^2 technique (they are classified as VII type in [26] and are similar to the example above). In [111], the adder and subtractor subcircuits on which the last three multipliers are based, are referred to as Fig. 1(a), (b), and (c), respectively. To be consistent with [111] while avoiding confusion with the figures in this thesis, we will refer to them as [111](a), (b), and (c), respectively. Table 6.3 shows a comparison of the IPI divider versus analog dividers. Sanchez-Sinencio et al. in [28] describe how to synthesize nonlinear functions including an analog divider from transconductance amplifiers (multipliers). Liu and Chang in [32] describe a divider based on the MOSFET square-law and the pool (current-equilibrium) circuits in [33]. Vlassis and Siskos in [34] describe a divider that consists of a voltage-variable resistor and a current conveyer that performs voltage and current follower operations. Unfortunately, we did not find analog implementations that use 0.35 um processes or a comparable process to compare our IPI multiplier with, except the ones in [111]. Therefore, before we make any comparison, we will discuss the impact of technology scaling on area, power, and speed.

	Error (%)	Speed (MHz) (Mop/s)	Average Power (mW)	Area (mm^2)	Dynamic range (mV)	Power supply (V)	CMOS Process (um)
IPI multiplier	*3*, 5	*100*, 10 *50*, 5	*6.72*	*51.8x10^{-3}*	600	3.2	0.35
[28]	3.5	1KHz Continuous	10	154x10^{-3}	660	±5	3
[29]	1	1KHz	N/A	N/A	±1000	±5	Discrete chips
[30]	0.89 @20KHz	*2.2 (-3dB)* *Continuous*	2.76	22.7x10^{-3}	440	1.2	0.80
[31]	2 @20KHz	*5 (-3dB)* *Continuous*	N/A	N/A	±800	±1.5	0.80
[111](a)	*17 (body)* *±16 (V_t)*	*N/A*	*N/A*	*N/A*	*200*	*3.3*	*0.35*
[111](b)	*27 (body)* *±10 (V_t)*	*48 (-3dB)* *Continuous*	*N/A*	*N/A*	*200*	*3.3*	*0.35*
[111](c)	*-5.8 (body)* *±16 (V_t)*	*84 (-3dB)* *Continuous*	*N/A*	*N/A*	*200*	*3.3*	*0.35*

Table 6.2. Comparison of the IPI multiplier versus analog multipliers. Results in *Italic* are from simulation. Errors in [111] are for the adder and subtractor subcircuits.

	Accuracy (%)	Speed (MHz) (Mop/s)	Average Power (mW)	Area (mm^2)	Dynamic range (mV)	Power supply (V)	CMOS Process (um)
IPI divider	*2*, 4	*100*, 10 *50*, 5	*3.36*	*25.9x10^{-3}*	600	3.2	0.35
[32]	*1*	*9 (-3dB)* *Continuous*	N/A	N/A	500	±5	2
[34]	5 @20KHz	*400 (-3dB)* *Continuous*	N/A	N/A	±1500	±2.5	2

Table 6.3. Comparison of the IPI divider versus analog dividers. Results in *Italic* are from simulation.

A good discussion of how technology scaling can impact area, power, and speed, can be found in [2]. Suppose S is the scaling factor (if greater than 1, the feature size decreases). Table 6.4 is a summary of the scaling relationships among device and circuit parameters. As transistor dimensions decrease by a factor of S, their area decreases by a factor of S^2. In examining the impact of scaling on power and speed, we will consider two cases. The first case is full scaling where all horizontal and vertical dimensions, as well as threshold voltages and supply voltage V_{DD}, are reduced by the same factor S. The second case is when supply voltage V_{DD} and threshold voltages are kept constant. Impact on power and speed can be found in a similar way if voltages are scaled by a factor different than S. DC power consumption is proportional to the product I_DV_{DD}. We can find from (6.9) and (6.10) that as oxide thickness t_{ox} decreases by a factor of S, the drain current I_D decreases by a factor of S for reduced V_{DD}, and increases by a factor of S for constant V_{DD}. This means that DC power consumption decreases by a factor of S^2 for reduced V_{DD}, and increases by a factor of S for constant V_{DD}. Speed is proportional to current and inversely proportional to capacitance and voltage swing (*Speed* α *I/CV*). From (6.10), as oxide thickness t_{ox} decreases by a factor of S, C_{ox} increases by a factor of S. But since the area of the gate is decreased by a factor of S^2, its capacitance is decreased by a factor of S. This means that speed increases by a factor of S for reduced V_{DD} and by a factor of S^2 for constant V_{DD}. Average AC power consumption of CMOS circuits is proportional to frequency (speed) in addition to the voltage swing and the average current charging the capacitive load (*average AC power* α $fI_{av}V$). The average current in turn is proportional to the load capacitance and the voltage swing. Therefore, the average AC power consumption is proportional to frequency, load capacitance, and the square of voltage swing (*average AC power* α fCV^2). This means that average AC power consumption decreases by a factor of S^2 for reduced V_{DD}, and increases by a factor of S for constant V_{DD}. This is similar to the conclusion we have reached for DC power consumption.

Parameter	Full scaling	Constant voltage
W, L, t_{ox}	$1/S$	$1/S$
V_{DD}, V_t	$1/S$	1
C_{ox}	$1/S$	$1/S$
I_D	$1/S$	S
Area	$1/S^2$	$1/S^2$
DC power consumption	$1/S^2$	S
AC power consumption	$1/S^2$	S
Speed	S	S^2

Table 6.4. Scaling relationships among CMOS device and circuit parameters.

When comparing our IPI multiplier with the analog multipliers in Table 6.2, we will discuss the issues surrounding local computation and the issues surrounding global (long wire) computation. When considering local computation, analog suffers from the following problems [26], [111], and [1]:

1. Body effect.
2. Device mismatch.
3. Mobility degradation.
4. Channel length modulation.
5. Velocity saturation in short-channel devices [1].

Excellent description and explanation of these problems can be found in [1].

The body effect is a main source of errors in analog computation. In CMOS circuits, the body of the transistor is usually connected to a constant voltage, which is the maximum voltage (V_{DD}) for PMOS, and the minimum voltage (V_{SS} or the ground) for NMOS. The reason for this is to avoid the possibility of forming forward-biased P-N junctions between the body and the drain or the source, to ensure correct operation. If the source voltage is changing while the body voltage is constant then V_{SB} is changing. Changes in V_{SB} will cause changes in the threshold voltage because V_{SB} changes the potential required to produce channel surface inversion, which is necessary for the channel to start conducting current. Therefore, this will cause changes in the drain current. From a small signal point of view, the body acts like a second gate. The body transconductance is typically about 0.1 to 0.3 of the main gate transconductance [1]. To explain how the body effect impacts the analog computation, we will use Figure 6.14. The figure shows the voltage adder used in the analog multiplier [111](b) to add the input voltages $X \pm x$ and $Y \pm y$ as in (6.11).

Let us first explain the ideal operation of the adder while ignoring all the problems described above. For ideal operation, each two transistors in series have to be identical. M1 has to be identical to M2 and M3 has to be identical to M4. Also, all of them have to be operating in saturation. Since the same current flows in M1 and M2 and they are identical and operating in saturation, their gate-source voltage is the same. This means

$$V_{d1} = V_{DD} - V_{gs2} = V_{DD} - V_1 \qquad (6.13)$$

By using the same reasoning for the PMOS pair, M3 and M4, and using (6.13), we can find

$$V_{out} = V_2 - V_{sg3} = V_2 - V_{sg4} = V_2 - (V_{DD} - V_{d1}) = V_2 + V_1 \qquad (6.14)$$

which is the ideal addition result we seek. However, M2 and M3 have the body effect problem because their source terminals are not connected to the ground and V_{DD}, respectively, as their body terminals. As we discussed above, this will change their threshold voltages. For NMOS, it will increase, and for PMOS, it will decrease (become more negative). This means their gate-source voltages are not equal to those of M1 and M4, respectively. For example, V_{gs2} needs to be higher than V_{gs1} so that M2 can produce the same current as M1 because its threshold voltage is higher than that of M1. This will make the addition deviate from its ideal operation and therefore cause some errors.

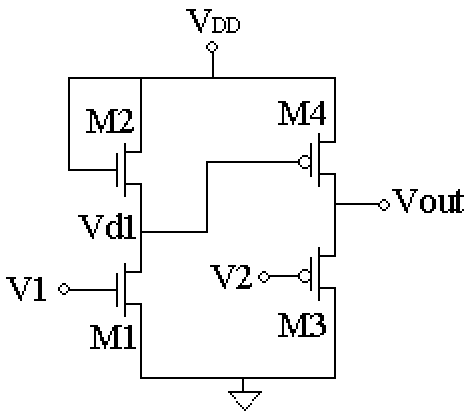


Figure 6.14. Analog voltage adder used in the analog multiplier [111](b).

Device mismatch due to process variations in the device parameters such as threshold voltage V_t, oxide thickness t_{ox}, channel width W, and channel length L, are also a main source of errors in analog computation. In (6.11), we assumed that all MOSFET transistors used for squaring, are identical and have the same scaling factor k. Mismatch in t_{ox}, W, or L will make our assumption invalid as (6.12) suggests. If we rewrite (6.11) without making that assumption then the new equation will be

$$z = \begin{bmatrix} \left(k_1[(X+x)+(Y+y)]^2 + k_2[(X-x)+(Y-y)]^2\right) \\ -\left(k_3[(X-x)+(Y+y)]^2 + k_4[(X+x)+(Y-y)]^2\right) \end{bmatrix} \qquad (6.15)$$

where k_1, k_2, k_3, and k_4 are the different scaling factors. From 6.15, we can see that undesirable terms may not cancel with each other because of the different scaling factors. The result will have terms that cause DC offset errors such as XY, X^2, and Y^2. It will also have terms that cause nonlinearity errors such as x^2 and y^2. Moreover, mismatch in the threshold voltages of the squaring transistors will also affect the voltage being squared, as (6.9) suggests, causing more errors. Device mismatch also causes errors in the addition and subtraction results. Consider our adder circuit example above. If two transistors in series (M1 and M2 or M3 and M4) are not identical then there will be errors in the addition result because their gate-source voltages are not equal.

Mobility degradation is another main source of errors in analog computation. When the vertical electrical field between the gate and channel increases, it forces the carriers closer to the surface of the silicon, where surface imperfections impede their movement from the source to the drain, reducing mobility [1] and [112]. For simplicity, this effect can be modeled by the following equation

$$\mu_{eff} = \frac{\mu_n}{1+\theta\left(V_{gs} - V_t\right)} \tag{6.16}$$

where μ_n is the mobility with zero vertical field, and θ is inversely proportional to the oxide thickness. For 100 Å oxide thickness, θ is typically in the range from 0.1 V^{-1} to 0.4 V^{-1}. Typical oxide thickness of a 0.4 um process is 80 Å [1].

Channel length modulation is another source of errors in analog computation. Ideally, if the transistor is in saturation, its current should stay constant even if the drain-source voltage V_{DS} changes. This ideal I-V relationship in saturation is described by (6.9). In practice, however, the drain-source voltage V_{DS} does modulate the current slightly. The physical explanation of this is that V_{DS} modulates the channel length by modulating the width of the depletion region between the drain and the channel pinch-off point [1]. To account for this effect, (6.9) is rewritten as

$$I = \mu C_{ox} \frac{W}{L}\left(V_{GS} - V_t\right)^2\left(1+\lambda V_{DS}\right) \tag{6.17}$$

where λ is called the channel length modulation parameter. It is difficult to calculate λ from the device structure and effective values of it are usually obtained from experimental measurements. λ is inversely proportional to the effective channel length. Therefore, channel modulation effects can be reduced by increasing the channel length at the expense of increasing the area and slowing the speed down [1] and [111]. Typical values of λ are in the range from 0.05 V^{-1} to 0.005 V^{-1} [1].

The most important short-channel effect in MOSFET transistors stems from the velocity saturation of carriers in the channel [113]. When V_{DS} is low and/or the channel is long, the horizontal electric field is low, and the relationship between the carrier velocity and the field is linear, leading to the square-law I-V characteristics defined in (6.8) and (6.9). At high field values, however, the carrier velocity approaches a constant called the scattering-limited velocity v_{scl}. This phenomenon causes the MOSFET I-V characteristics to deviate from the classical square-law characteristics and to be more linear. Therefore, in processes with 1 um or less capability, many transistors in analog computation circuits may need to be deliberately designed to have lengths larger than the minimum, so they can be approximated by the square-law models [1]. This effect and the channel length modulation effect are clear examples on how analog computation circuits do not scale well in submicron, deep submicron, and nano technologies.

We are now ready to compare our IPI multiplier with the analog multipliers in Table 6.2. We will focus the comparison on the three multipliers in [111] for the following two reasons:

1. The multipliers in [28]-[31] are implemented using a different process.
2. The reported nonlinearity error in [28]-[31] is very small, but unfortunately the experimental results do not have data on how accuracy changes with respect to body effect, device mismatch due to process variations, mobility degradation, or channel length modulation. Reference [111] does have such data at least for the body effect and threshold voltage mismatch.

From the table, we can see that our IPI multiplier is, in general, comparable to the analog multipliers in terms of area and power consumption. However, as discussed above, due to the channel length modulation effect and the carrier velocity saturation effect, the analog multipliers do not scale well in advanced processes, and cannot take advantage of the small area and the high speed their short-channel devices can offer. On the other hand, the IPI multiplier and the other IPI circuits have the channel length modulation problem only in the current sources that charge the capacitors. Therefore, we have used long-channel devices for these current sources. All the other transistors do not have the channel length modulation problem, and therefore we were able to use the minimum length for these transistors and take advantage of the small area and the high speed they can offer. For the adder subcircuits used in the multipliers [111](a) and [111](b) and the subtractor subcircuit used in the multiplier [111](c), the body effects accounted for percentage errors of 17%, 27%, and -5.8%, respectively, in the output voltage of each subcircuit from its ideal value. Body effect is not a problem in our IPI circuits because the source terminals of the PMOS and NMOS current sources are connected to their body terminals, which are connected to V_{DD} and V_{SS} (ground), respectively. Body effect can be reduced by using a twin-well process that has isolated wells and allows the transistor source to be connected to its well (body) [114] and [115]. These extra processing steps are not needed for our IPI circuits. Using mismatch data in V_t provided by the foundry, worst case mismatch errors in the output voltage accounted for about ±16% for [111](a) and [111](c), and about ±10% for [111](b). In the IPI circuits, if there is an inaccuracy or a mismatch in the capacitors or the current sources which generate the currents that charge the capacitors, then they will cause a scaling error in the output. Whereas in analog multipliers, device mismatch causes DC offset and nonlinearity errors in addition to the scaling error. If an exact scaling factor is desired then scaling errors in the IPI circuits can be minimized by calibration, as we will discuss in section 7.2. As we explained above, mobility degradation is a primary source of errors in analog computation. Our IPI circuits do not have the mobility degradation problem because the IPI inputs only control the ON/OFF switching of the input transistors and do not modulate the magnitudes of the currents passing through them. The currents magnitudes are decided by the current sources, which they do not have the mobility degradation problem because their gate-source bias voltages (their vertical fields) are fixed.

With regard to minimum power supply requirements, several low-voltage analog multipliers have been reported [30] and [31]. As examples, the multipliers in [30] and [31] can operate from 1.2 V supply and ±1.5 V (3 V total) supplies, respectively, while achieving a good dynamic range of 440 mV and ±800 mV, respectively. The multiplier in [30] achieves a lower supply voltage because it uses only one stack of transistors with resistors while the one in [31] uses two stacks with resistors. Our IPI multiplier operates from 3.2 V (±1.6 V). Our IPI circuits do not have many stacks of transistors between the power rails. There are only two transistors between V_{DD} (3.2 V) and V_{middle} (1.6 V). One serves as a current source and the other serves as an ON/OFF switch. The ON/OFF switch can be sized and strongly turned on so that its voltage drop is minimized. This is almost like having only one transistor (the current source) between V_{DD} (3.2 V) and V_{middle} (1.6 V). Therefore, our IPI circuits should be able to run at a voltage lower than 3.2 V, although in our work, we have chosen 3.2 V which is close to the TSMC foundry recommendation of 3.3 V [118]. Optimizing the IPI circuits for low-voltage operation in deep submicron technologies such as 0.18 um or 0.13 um or even nano technologies is a promising area for future research.

When considering issues surrounding global long-wire computation, IPI has all the local computation advantages above, as well as the advantages of communication since the computation involves communication of the signals over a relatively long distance (long wire). As we have discussed in the introduction chapter, the main advantage of IPI in communication is its immunity to noise [6], [7], [16], [101], and [102], process variations, temperature, and reference voltage, and its immunity to the problems that challenge complex mixed-signal SOC integration in deep submicron and nano technologies, such as substrate coupling, cross-talk, transmission line effects, threshold inconsistency, subthreshold currents, hot-electron effects, and doping variability [21] and [64-68]. The reason is that IPI encodes the information using the time between the pulses rather than their magnitude. This is basically converting the analog information to carefully timed signal transitions that are similar to digital schemes. A pulse is detected if it is above a certain voltage threshold, exactly in the same way a binary 0 or 1 value is detected in the on-off digital scheme [105]. Because of this also, pulses can be easily transferred and refreshed using digital buffers, unlike analog signals which are sensitive to noise and degrade in magnitude especially if they need to travel over a relatively long distance. This makes pulses a much better choice for inter-chip communication [6] and [7], or even for transferring the signal within the same chip if the wire is relatively long or noise or cross-talk, for example, is a concern as in SOC.

Switched-capacitor (SC) circuits are clocked sampled-data analog systems, and therefore they occupy an intermediate position between fully analog (continuous-time/continuous-amplitude) and fully digital (discrete-time/discrete-amplitude) systems [127]. In analog signal processing, SC filters have certain advantages over active RC filters. The most important advantages are the accuracy and the center (cut-off) frequency tunability [128] and [131]. These advantages come from the ability of the SC circuits to simulate the resistor element needed in active RC filters using an on-chip capacitor and two switches such that the value of the simulated resistor is

$$R = \frac{T}{C} \tag{6.18}$$

where T is the cycle time of the clock controlling the switches. A time constant, say $\tau_2=R_1C_2$ is then equal to

$$\tau_2 = R_1 C_2 = T\frac{C_2}{C_1} \tag{6.19}$$

From (6.19), the cut-off frequency of the SC filter can by easily and accurately tuned using only an external accurate clock with no need for external components such as resistors or capacitors. Also from (6.19), the cut-off frequency is dependent on the ratio of the capacitor values (not the values themselves), which is very accurate especially if the capacitors are placed close to each other [128]. SC circuits can also be used to implement other operations such as ADC, DAC, integration (accumulation), and amplification [128]. Some multipliers based on the SC techniques have also been reported [124-126]. Similar to the continuous analog multipliers described above, they rely on identical CMOS transistors operating in the triode region [124] and [125] or the saturation region [126]. They also use subtraction to cancel undesired DC and nonlinear terms. Therefore, device mismatch and deviation from the square-law I-V MOSFET characteristics will cause DC offset and nonlinearity errors. The multiplier in [126] does not have the body effect problem but the multipliers in [124] and [125] do. All of them have the mobility degradation problem. Our IPI implementations and SC implementations both need to use capacitor values large enough to reduce inaccuracy due to parasitic capacitances. IPI is comparable to SC in terms of area and power consumption since they both use capacitors, switches, and OP-AMPs, and with comparable numbers. However, what makes our IPI implementations different from the SC implementations is that the inputs to the SC circuits are voltages and the outputs are also voltages. Therefore, SC circuits, as continuous analog circuits, are not suitable for long-wire or inter-chip communication or computation, and the signal has to be converted to digital or pulses for transmission if high noise immunity is needed.

6.2.2. Comparison with Digital Implementations

It is important here to remember that the IPI representation is mainly intended for signals that are originally analog not digital. So, it is not completely fair to the IPI computation if we compare it against other digital computations while forgetting that digital computation needs analog-to-digital conversion (ADC) before they can have their digital inputs. So, we will first compare the V-to-IPI conversion against some ADC implementations.

There are multiple techniques for ADC depending on the speed, accuracy (or bit resolution), area, power and other design requirements. Table 6.5 is a comparison of the V-to-IPI converter versus various ADC implementations. The 1-step full-flash ADC like [35] is the fastest but the most expensive in terms of area and power since it needs 2^{N-1} comparators for N-bit resolution. The folding and current interpolating ADC like [36]-[39] achieves high speed but with less number of comparators. The 2-

step flash ADC like [40] also achieves a relatively high speed with less area and power than the 1-step full-flash ADC by the use of subranging (two ADCs are used, one for the coarse bits and another for the fine bits). Test results of the V-to-IPI show accuracy better than 96% at 10 MHz and SPICE simulation results show accuracy better than 98% at 100 MHz. This is equivalent to 5.6-bit digital accuracy.

From the results in table 6.5, we can see that the ADC with the least area and power consumption at 3.3 V power supply occupies 59 times the area and consumes 36 times the power of our V-to-IPI converter. Its sampling speed is comparable, 80 MS/s versus 100 MS/s. However, its resolution is 8-bit versus 5.6-bit. The results in the table show that our V-to-IPI converter is very compact and consumes very low power compared to all ADCs. This makes it a significantly better choice than ADC for applications such as sensors, instrumentation, communications, and telemetry if 98% accuracy, which is equivalent to 5.6-bit accuracy, is adequate.

	Resolution (bits)	Speed (MS/s)	Average Power (mW)	Area (mm^2)	Power supply (V)	CMOS Process (um)
V-to-IPI	98%(5.6)	100	2.24	$5.12x10^{-3}$	3.2	0.35
[35]	6	500	225	0.8	3.3	0.35
[36]	8	80	80	0.3	3.3	0.50
[37]	8	200	210	0.96	3.0	0.35
[38]	6	50	20	4.8	1.0	0.35
[39]	7	300	200	1.2	3.3	0.35
[40]	10	25	195	0.66	3.3	0.35

Table 6.5. Comparison of the V-to-IPI converter versus ADC implementations.

Table 6.6 is a comparison of the IPI multiplier versus some digital implementations: a pipelined multiplier [41], a multiplier based on redundant-addition with improved redundant-binary to normal-binary conversion of the final result [42], a full-array multiplier [43], and a look-up table estimated based on the 64M bit DRAM implementation in [44]. The size of the look-up table is 4K x 12-bit. The inputs are each 6-bit and the output is 12-bit. The area and power for the look-up table were calculated from the area and power results of the 64 M bit DRAM in [44] by simple averaging. The access time and cycle speed are difficult to estimate but they will be much faster for the look-up table since it needs a much smaller address decoder. We have chosen 6-bit resolution for the look-up table so it is comparable to the equivalent resolution of the IPI multiplier (97% is equivalent to 5.6-bit resolution). The actual 64M bit DRAM has a 20 ns access time for 3.3 V power supply and draws 57 mA of current at 80 ns cycle time. The memory cell is a stacked type capacitor with a cell size of 3.04 um^2.

	Resolution (bits)	Speed (MHz)	Mega operations per second (Mop/s)	Average Power (mW)	Area (mm^2)	Power supply (V)	CMOS Process (um)
IPI multiplier	Eqiv. to 5.6x5.6	100	33.3	6.72	51.8x10^{-3}	3.2	0.35
[41]	8x8	300	300 if full pipeline	52.4 *36.6*	N/A	3.3	0.6
[42]	54x54	100	100	540 *5.8*	9.4 *101x10^{-3}*	3.3	0.50
[43]	54x54	100	100	870 *9.4*	12.49 *134x10^{-3}*	3.3	0.50
Look-up table estimate based on DRAM in [44]	6x6 (input) 12 (output)	>50	>50	0.6 @80ns cycle	304x10^{-3}	3.3	0.50
Multiply-and-Accumulate[45]	32x32 64(Accum)	56.5	56.5	330	2.35	2.9	0.40
Multiply-and-Accumulate[46]	12x12 27(Accum)	200	200	1300	9.25	5	1

Table 6.6. Comparison of the IPI multiplier versus digital multipliers. Area and power results in *Italic* are estimates for an equivalent 5.6-bit x 5.6-bit multiplier.

When comparing the results in the table, we should remember that the area and power consumption of digital multipliers (except sequential multipliers which use one adder repeatedly), increase exponentially with the bit resolution. For example, a 16x16 full-array multiplier needs area and power 4 times more than an 8x8 multiplier of the same type. So, we need to keep this consideration in mind to be fair to the high resolution digital multipliers in the table. The area and power results in *Italic* are estimates if the high resolution multipliers are shrunk to a smaller multiplier of hypothetical size of 5.6-bit x 5.6-bit. We can conclude from the results in the table that digital multipliers in general are comparable in terms of power and area to the IPI multiplier. The look-up table DRAM solution however consumes the least power of all, about 10 times less power than the IPI multiplier. As we emphasized in the beginning, judging between the IPI solution and any digital solution should be based on the total cost of conversion and computation and not only on one of them. As we have found above, IPI conversion is much less expensive than the least expensive ADC. So, the IPI solution is better than the digital in applications where 98% accuracy (5.6-bit resolution) is adequate. This is particularly true if the number of analog inputs (conversions from analog) is large to the point that the cost of ADCs outweighs the cost of computations. These conclusions are also valid for the other arithmetic functions: division, addition, and subtraction.

The fan-out of our IPI circuits is large, as in digital CMOS, since the pulse output drives CMOS logic gates. However, in both IPI and digital, the output rise and fall times increase as the number of inputs the output is driving increases because the load interconnect and gate capacitances increase. The fan-in of our IPI circuits is also large since each input has its own gates and switches. This makes them very scalable to any number of inputs within the dynamic range of operation, of course.

7. Conclusions and Future Work

In this Chapter, we will summarize the importance of this work and discuss future areas of research.

7.1. Conclusions

Analog signal representation is essential wherever there is a need to interface with the analog world or to satisfy certain design requirements such as power consumption, area, or speed. Analog signal representation is also useful when integrating analog mixed-signal and RF functions into complex SOCs. Moreover, data is usually obtained from sensors in analog form (voltage or current). The analog signals are not immune to noise and therefore cannot be used for data transfer. Therefore, they are usually converted into digital immediately and transmitted digitally. This whole path also has an inverse which would be used, for example, in actuators.

In this thesis, we have proposed an alternative approach that converts the analog signal into a pulse stream, using time (IPI) rather than magnitude to represent the signal values. Our approach is suitable and robust in both conversion/communication and computation. Its capabilities in both conversion/communication and computation are useful because they eliminate the need to convert to/from other analog or digital domains for computation, when needed. One good example where computation would be needed with communication is the use of averaging at the front end of the receiver to improve the SNR. We showed that our approach is hybrid in that it blends noise immunity of digital with the compactness and low power consumption of analog.

Other pulse time (IPI and PWM) representations where reviewed in this thesis. We showed that our approach is more suitable when negative, zero, and positive values are needed, because our representation uses both parts of the frame (t_+ and t_-) not just one part, which eliminates the need to keep track of offsets during computation. The other approaches can convert to analog or digital and use their techniques for computation but each method has its own disadvantages. Considering only local analog computation, it suffers from serious problems such as the body effect and the mobility degradation effect. Choosing digital computation means that we have to convert from these time representations into digital, which requires a very fast counter and clock assuming that their conversion speed is as good as ours. This is not practical and therefore speed has to be significantly slowed down to accommodate the digital counter operation. Our approach allows for both conversion/communication and computation in the same time domain and at comparable speeds.

Different conversion schemes, linear and nonlinear (logarithmic), and also different signaling schemes, synchronous and asynchronous, were investigated. Our representation is linear and synchronous. The linear scheme provides better accuracy, is less complex to realize in CMOS, and is more suitable for computation than the logarithmic scheme. These advantages outweighed the advantage of higher bandwidth (speed) due to time compression in the logarithmic scheme. The synchronous scheme is more suitable for computation than the asynchronous scheme, especially on negative values. This advantage outweighed the advantage of higher bandwidth (speed) coming from the fact that the asynchronous scheme does not allocate a full time frame for each value as the synchronous scheme does.

We demonstrated the feasibility of our novel IPI representation in both conversion/communication and computation by developing a class of novel methods and circuits for basic conversion and computation based on it. These methods and circuits include IPI-to-V conversion, V-to-IPI conversion, and the basic computations:

addition, subtraction, division, and multiplication. These methods and circuits were successfully demonstrated through mathematical derivations, complex BSIM3v3.1 SPICE simulations, and chip design, fabrication, and test using the TSMC 0.35 um mixed-signal CMOS fabrication process technology. They were simulated and tested at a 10 MHz framing speed (100 ns time frame). Test and simulation results agreed with the calculated results. We have also simulated the conversion, addition, and subtraction circuits at 100 MHz, and the division and multiplication circuits at 50 MHz, and simulation results agreed with the calculated results. To demonstrate the operation of these basic IPI blocks together at the system-level, a 6-synapse neuron was designed and simulated.

Simulation results have shown similar accuracy and dynamic range at both low and high speeds. Accuracy of more than 98%, low power consumption of less than 2.1 mW, small area of less than $7.68\text{x}10^{-3}$ mm^2, and a wide dynamic range of 1200 mV were achieved for conversion, addition, and subtraction. For division, accuracy of more than 98%, low power consumption of 3.36 mW, small area of $25.92\text{x}10^{-3}$ mm^2, and a dynamic range of 600 mV were achieved. And for multiplication, accuracy of more than 97%, low power consumption of 6.72 mW, small area of $51.84\text{x}10^{-3}$ mm^2, and a dynamic range of 600 mV were achieved.

This thesis also discussed how the IPI technology can be used in applications such as sensors, instrumentation, communications, telemetry, signal processing, and ANNs. We have compared our IPI based conversion and computation implementations against other analog and digital implementations and tied the results with the applications. Our IPI implementations are comparable to the more traditional analog implementations in terms of area and power consumption. However, they are more robust than the analog implementations studied here. In communication whether on-chip or off-chip, the main advantage of the IPI over analog is its digital-like immunity to noise, process variations, temperature, reference voltage, and other serious problems such as cross-talk and substrate coupling, because of the fact the IPI uses time between pulses rather than magnitude to represent the analog signal.

Issues surrounding local analog computation were discussed. Our IPI implementations do not have the serious problems that analog computation suffers from such as the body effect, the mobility degradation effect, the offset and nonlinearity errors due to device mismatch, and the unsuitability for scaling in short-channel deep submicron and nano technologies due to deviation from the MOSFET I-V square-law caused by carrier velocity saturation.

When compared with digital implementations, V-to-IPI conversion is significantly less expensive in terms of area and power consumption than the least expensive ADC. Therefore, V-to-IPI is a significantly better choice than ADC when 98% (5.6-bit) accuracy is adequate. IPI computation is comparable to digital computation in terms of area and power consumption but the look-up table DRAM solution consumes the least power among all. Therefore, the decision of using IPI versus digital for computation should be based on the total cost of conversion and computation.

7.2. Future Work

One possible speed optimization that one can try in the future is to use pipelining to overlap the sampling and conversion operations so that T_{max} and T_{sd} are no longer additive and T_f then needs to satisfy the relationship $T_f > max\ (T_{sd}\ ,\ T_{max})$ instead of the relationship $T_f > (T_{sd} + T_{max})$. The V-to-IPI conversion circuitry described in section 4.1 can be modified to implement this capability by the use of a second capacitor. The two capacitors interchange their roles in each frame. In one frame, the first capacitor is used to sample the signal while the other capacitor is used to convert the previous sample. In the next frame, the first capacitor is used to convert its sample while the second capacitor is used to get a new sample of the signal, and so on. This should improve the speed by a factor of

$$K_{speed} = \left(\frac{speed_2}{speed_1} - 1 \right) \times 100\% = \left(\frac{T_{f1}}{T_{f2}} - 1 \right) \times 100\% = \left(\frac{T_{sd} + T_{\max}}{\max\left(T_{sd}, T_{\max}\right)} - 1 \right) \times 100\% \qquad (7.1)$$

The cost of this speed gain is, of course, the extra area and power needed for the second capacitor and the logic gates and the switches that will be required to control the interchange of the two capacitors.

Another possible speed optimization is to use pipelining to overlap the input conversion and output pulse generation steps needed in computation, which would apply to all of the computation circuits in chapter 4. In the case of division, for example, this can be done by using a second set of capacitors. The two sets interchange their roles in each frame. In one frame, C_1 and C_2 of the first set are used for input conversion while the second set is used to generate the output pulse. In the next frame, C_1 and C_2 of the second set are used for input conversion while the first set is used to generate the output pulse, and so on. This should double the speed. As for V-to-IPI conversion, the cost of this speed gain is the extra area and power needed for the second set of capacitors and the logic gates and switches that will be required to control the interchange of the two sets.

While developing and designing our IPI methods/circuits, our main design concern or target was functionality. Once we had the basic designs, we then started optimizing them for accuracy, then for area and power consumption so that our IPI circuits are comparable with their analog counterparts. During this process, we found that their dynamic range and speed can be very good. Speed was not among our top priorities because we were initially designing these circuits for communication and signal processing of analog sensory data in the low to medium speed range [122]. Further optimization of our IPI circuits is possible and should be investigated in future work. Consider the following equation which describes the voltage of a capacitor *C* being charged by the current *I*. As we have seen in chapter 4, this equation is at the heart of each of our IPI circuits.

$$V = \frac{I}{C} t \qquad (7.2)$$

As we have also explained in chapter 4, the dynamic range of operation of our IPI circuits is from $-V_M$ to $+V_M$ where V_M is equal to

$$V_M = \frac{I}{C} T_f \qquad (7.3)$$

Equations (7.2) and (7.3) tell us how to optimize our circuits for area, power consumption, speed, or dynamic range. Speed is inversely proportional to the frame time T_f. If we rearrange (7.3), we get

$$T_f = \frac{C}{I} V_M \qquad (7.3)$$

We can improve speed by doing one or more of the following:

1. Decreasing C: This will also decrease the area needed by C. However, we should not make C very small because this will increase the ratio of parasitic capacitance to desired capacitance and consequently decrease accuracy [123].
2. Increasing I: this will increase the power consumption. It will also increase the area needed by the current sources.
3. Decreasing V_M: This means decreasing the dynamic range of operation.

Increasing speed by decreasing C will lead to a much smaller increase in the power consumption than increasing it by increasing I. The reason is that using the first method, the current I stays fixed and the small increase in power consumption will come from the fact that the CMOS dynamic logic will be switching more often because of the higher speed. CMOS logic consumes power mainly during switching and power consumption is almost zero during steady state [2]. The very small steady-state power consumption is due to leakage currents. Therefore, most of the power consumption is coming from the sources charging the capacitors. While this suggests that it is better to increase speed by decreasing C and not by increasing I, we cannot make C too small since that will degrade accuracy.

Equation (7.3) also suggests that we can reduce both area and power consumption while keeping the same speed and dynamic range. This can be done by decreasing both C and I by the same factor, so that the ratio I/C is constant. The savings in area will come from the smaller capacitance and the smaller current sources. The savings in power consumption will come from the smaller currents needed to charge and discharge these smaller capacitors.

There can also be some area and power savings that may be achieved by making the reset switches smaller but this will widen the pulse generated by the comparator and the reset circuitry. This optimization depends on how short the generated pulse needs to be and how much time it can take from the frame.

Another important suggestion for future work is to implement a calibration scheme for the IPI circuits. As we have explained in subsection 6.2.1, process variations in capacitors and current sources and the reference voltage variations cause scaling and offset errors, respectively. Therefore, a calibration scheme is crucial to the maximum accuracy of the IPI circuits. As a simple example, on-die reference voltage can be obtained by a voltage divider. A simple voltage divider can be implemented using two or more diode-connected transistors connected in series between V_{DD} and V_{SS} (or the ground). A diode-connected CMOS transistor does not actually behave like a diode. "Diode-connected" is just a term that is used to describe a CMOS transistor whose drain and gate are connected to each other like a diode-connected bipolar transistor whose collector and base are connected to each other to operate it as a diode [1]. Suppose that V_{REF} is some reference voltage that we take from the source or the drain of one of these transistors in series. These transistors operate as resistors. We can calibrate V_{REF} by varying their resistance. We can vary their resistance by connecting more or less transistors in parallel to them. Connecting or disconnecting these transistors can be done digitally. For automatic calibration, some counter and decoder will be needed to decide how many and which transistors should be connected in order to minimize the offset error. Automatic calibration of current sources and capacitors can be done in a very similar way: Connect more or less of them until the error is minimum.

The choice of whether to calibrate I or C depends on, of course, area and power optimization targets, as we explained above. Of course, calibration requires that we provide the chip with the expected output in order for the calibration circuit to calculate the error and come up with the digital numbers that give the minimum error (the maximum accuracy). High resolution calibration will require higher digital accuracy and therefore more area and power consumption. Calibration can be done once or periodically depending on whether there are other time-dependent variations that it needs to account for or not, such as temperature for example. As we can see from this discussion, a calibration scheme can be also exploited to serve as a programming scheme that can be used to program the current sources and/or the capacitors so that the chip can run at different speed and power consumption levels. Coarse settings can be used for programming the chip for different speed and power consumption levels while fine settings can be used to calibrate for maximum accuracy. Such an implementation is basically similar to an analog field programmable array (FPGA) implementation.

As we have explained in subsection 6.2.1, our IPI circuits should be able to run at low-voltage because they do not have many stacks of transistors. In this work, we use 3.2 V supply voltage which is close to the TSMC foundry recommendation of 3.3 V [118]. Optimizing the IPI circuits for low-voltage operation in deep submicron technologies such as 0.18 um or 0.13 um or even for nanoscale circuits is an area of potential research.

In chapter 4, we described a simple scheme for receiving the synchronous IPI inputs and converting them to local square wave (PWM) signals which carry the same t_+ and t_- IPI information. The scheme uses toggle flip-flops. The problem with this method is that if a pulse is lost or a spurious pulse is received then the T-FF will be stuck in an incorrect phase and all subsequent IPI values will be erroneous. Another promising topic for future work is to use a more sophisticated clock-recovery scheme such as phase-locked loop (PLL) [120] to address this reliability problem with synchronous IPI signaling. Such a problem does not exist in asynchronous IPI signaling since a lost pulse or a spurious pulse will cause a maximum of two errors only.

Bibliography

1. Paul R. Gray, Paul J. Hurst, Stephen H. Lewis, Robert G. Meyer, Analysis and Design of Analog Integrated Circuits, 4th ed., New York: Wiley, 2001.

2. David A. Hodges, Horace G. Jackson, Analysis and Design of Digital Integrated Circuits, 2nd ed., New York: McGraw-Hill, 1983.

3. John F. Wakerly, Digital Design Principles and Practices, 2nd ed., Englewood Cliffs, NJ: Prentice-Hall, 1994.

4. M.A. Jabri, R.J. Coggins, and B.G. Flower, Adaptive Analog VLSI Neural Systems, London, England: Chapman and Hall, 1996.

5. Carver A. Mead, Analog VLSI and Neural Systems, Reading, MA: Addison-Wesley, 1989.

6. A. F. Murray, D. Del Corso, and L. Tarassenko, "Pulse-Stream VLSI Neural Networks Mixing Analog and Digital Techniques," IEEE Trans. Neural Networks, vol. *2,* no. 2, March 1991, pp. 193-204.

7. Hamilton, A., Murray, A.F., Baxter, D.J., Churcher, S., Reekie, H.M., and Tarassenko, L., "Integrated pulse stream neural networks: results, issues, and pointers," IEEE Transactions on Neural Networks, vol. 3, no. 3, May 1992, pp. 385-393.

8. A.F. Murray et al., "Pulse stream VLSI neural networks," IEEE Micro, vol. 14, no. 3, June 1994, pp. 29-39.

9. Brownlow, M.J., Tarassenko, L., and Murray, A.F., "Analogue computation using VLSI neural network devices," Electronics Letters, vol. 26, no. 16, Aug. 1990, pp. 1297-1299.

10. Woodburn, R. and Murray, A.F., "Pulse-stream techniques and circuits," IEEE Circuits and Devices Magazine, vol. 12, no. 4, July 1996, pp. 43-47.

11. A.F. Murray and A.V.W. Smith, "Asynchronous arithmetic for VLSI neural systems," Electorn. Lett., vol. 23, no. 12, June, 1987, pp. 642-643.

12. A.F. Murray and A.V.W. Smith, "A novel computational and signaling method for VLSI neural networks," in Proc. European Solid State Circuits Conf., 1987, pp. 10-22

13. Murray, A.F. and Smith, A.V.W., "Asynchronous VLSI neural networks using pulse-stream arithmetic," IEEE Journal of Solid-State Circuits, Volume 23, Issue 3, June 1988, pp. 688–697.

14. P.B. Denyer and J. Mavor, "MOST transconductance multipliers for array applications," Proc. Inst. Elec. Eng., pt. 1, vol. 128, no. 3, June 1981, pp. 81-86.

15. Han, I.S. and Park, S.B., "Voltage-controlled linear resistor by two MOS transistors and its application to active RC filter MOS integration," Proceedings of the IEEE, Volume 72, Issue 11, Nov. 1984, pp. 1655–1657.

16. Del Corso, D. and Reyneri, L.M., "Mixing analog and digital techniques for silicon neural networks," IEEE International Symposium on Circuits and Systems, vol. 3, 1990., 1-3 May 1990, pp. 2446-2449.

17. D. Del Corso, F. Gregoretti, C. Pellegrini, and L.M. Reyneri, "A pulse stream synapse based on a closed loop register," in Proc. Third Int. Conf. Parallel Archeictures and Neural Networks (Vietri sul Mare, Italy), May 1990.

18. D. Del Corso, F. Gregoretti, and L.M. Reyneri, "Use of pulse rate and width modulations in mixed analog digital cell for artificial neural systems," in Proc. NATO ARW Neuro computing, Algorithm, Archetictures and Applicatoins (Les Arcs), Feb. 1989.

19. E. Culurciello and A. G. Andreou, "ALOHA CMOS imager," in Proceedings of the 2004 IEEE International Symposium on Circuits and Systems, ISCAS '04, May 2004.

20. Teixeira, T., Andreou, A.G., and Culurciello, E., "Event-based imaging with active illumination in sensor networks," IEEE International Symposium on Circuits and Systems, vol. 1, 23-26 May 2005, pp. 644-647.

21. D. Hammerstrom, M. Jabri, and R. Etienne-Cummings, "Inter-Pulse-Interval Based Mixed Signal Representations," Research proposal submitted to the Semiconductor Research Corporation, 2001.

22. S. Ravi and D. Hammerstrom, "Inter-Pulse-Interval Analog Signal Representation for Low Power Sensor Data Collection," In Preparation.

23. D. Clein, CMOS IC Layout: Concepts, Methodologies, and Tools, 1st ed., Woburn, MA: Newnes, 1999.

24. R. Jacob Baker, CMOS: Circuit Design, Layout, and Simulation, 1st ed., New York: Wiley, 1997.

25. R. Jacob Baker, CMOS Mixed-Signal Circuit Design, 1st ed., New York: Wiley, 2002.

26. G. Han and E. Sanchez-Sinencio, “CMOS transconductance multipliers: A tutorial,” IEEE Trans. Circuits Syst. II, vol. 45, pp. 1550–1563, Dec. 1998.

27. B. Gilbert, “A precision four-quadrant multiplier with subnanosecond response,” IEEE J. Solid-State Circuits, vol. SC-3, pp. 353–365, Dec. 1968.

28. E. Sanchez-Sinencio et al., “Operational transconductance amplifier-based nonlinear function syntheses, IEEE J. Solid-State Circuits, vol. 24, Dec. 1989.

29. S. Liu and Y. Hwang, “CMOS four-quadrant multiplier using bias feedback techniques,” IEEE J. Solid-State Circuits, vol. 29, pp. 750–752, June 1994.

30. Shuo-Yuan Hsiao and Chung-Yu Wu, “A parallel structure for CMOS four-quadrant analog multipliers and its application to a 2-GHz RF downconversion mixer,” IEEE Journal of Solid-State Circuits, vol. 33, no. 6, June 1998, pp. 859-869.

31. S.-I. Liu and C.-C. Chang, “Low-voltage CMOS four-quadrant multiplier,” Electron. Lett., vol. 33, no. 3, Jan. 1997, pp. 207–208.

32. S. Liu and C. Chang, “CMOS analog divider and four-quadrant multiplier using pool circuits,” IEEE J. Solid-State Circuits, vol. 30, pp. 1025–1029, Sept. 1995.

33. Tsay, S.W. and Newcomb, R.W., "A neural-type pool arithmetic unit,” IEEE International Sympoisum on Circuits and Systems, vol. 5, 11-14 June 1991, pp. 2518–2521.

34. Vlassis, S. and Siskos, S., "Analog CMOS four-quadrant multiplier and divider,” Proceedings of the 1999 IEEE International Symposium on Circuits and Systems, Volume 5, 30 May-2 June 1999, pp. 383 – 386.

35. I. Mehr and D. Dalton, "A 500-MSample/s, 6-Bit Nyquist-Rate ADC for Disk-Drive Read-Channel Applications,” IEEE J. Solid-State Circuits, vol. 34, no. 7, July 1999.

36. A. G. W. Venes and R. J. van de Plassche, “An 80-MHz, 8-b CMOS folding A/D converter with distributed track-and-hold preprocessing,” IEEE J. Solid-State Circuits, vol. 31, pp. 1846–1853, Dec. 1996.

37. S. Kim and M. Song, “An 8-b 200 MSPS CMOS A/D converter for analog interface module of TFT-LCD driver,” in Proc. IEEE Int. Symp. Circuits and Systems, 2001, pp. 528–531.

38. B. Song, P. Rakers, and S. Gillig, “A 1-V 6-b 50-Msamples/s current interpolating CMOS ADC,” IEEE J. Solid-State Circuits, vol. 35, pp. 647–651, Apr. 2000.

39. Y. Li and E. Sánchez-Sinencio, "A Wide Input Bandwidth 7-bit 300-MSample/s Folding and Current-Mode Interpolating ADC," IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 38, NO. 8, AUGUST 2003.

40. H. van der Ploeg and R. Remmers, "A 3.3-V, 10-b, 25-MSample/s Two-Step ADC in 0.35- m CMOS," IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 34, NO. 12, DECEMBER 1999.

41. J. Wang, P. Yang, and D. Sheng, "Design of a 3-V 300-MHz Low-Power 8-b X 8-b Pipelined Multiplier Using Pulse-Triggered TSPC Flip-Flops," IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 35, NO. 4, APRIL 2000, pp. 583-592.

42. H. Makino et al., "An 8.8-ns 54 x 54-Bit Multiplier with High Speed Redundant Binary Architecture," IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 31, NO. 6, JUNE 1996, pp. 773-783.

43. J. Mori et al., "A 10-ns 54 x 54-b Parallel Structured Full Array Multiplier with 0.5-pm CMOS Technology," IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 26, NO. 4, APRIL 1991, pp. 600-606.

44. M. Agata et al., "A Circuit Technology for High-speed Battery-Operated 16-Mb CMOS DRAM’s," IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 28, NO. 11, NOVEMBER 1993, pp. 1084-1091.

45. H. Murakami et al., "A Multiplier-Accumulator Macro for a 45 MIPS Embedded RISC Processor," IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 31, NO. 7, JULY 1996, pp. 1067-1071.

46. F. Lu and H. Samueli, "A 200-MHz CMOS Pipelined Multiplier-Accumulator Using a Quasi-Domino Dynamic Full- Adder Cell Design," IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 28, NO. 2, FEBRUARY 1993, pp. 123-132.

47. J. Lazzaro, S. Ryckebusch, M.A. Mahowald, and C.A. Mead. Winner-take-all networks of o(n) complexity. NIPS, 1, 1988.

48. Abrahamsen, J.P., Hafliger, P., and Lande, T.S., "A time domain winner-take-all network of integrate-and-fire neurons,” Proceedings of the 2004 International

Symposium on Circuits and Systems, Volume 5, 23-26 May 2004, pp. V-361 - V-364.

49. M. Mahowald, "VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function," Ph.D. Thesis, Computation and Neural Systems, California Institute of Technology, 1992.

50. A. Mortara and E.A. Vittoz, "A Communication Architecture Tailored for Analog VLSI Artificial Neural Networks: Intrinsic Performance and Limitations," IEEE Trans. Neural Networks, vol. 5, no. 3, May 1994, pp. 459 - 466.

51. E. Culurciello, R. Etienne-Cummings, and K. A. Boahen, "A biomorphic digital image sensor," IEEE Journal of Solid-State Circuits, vol. 38, no. 2, February 2003, pp. 281–294.

52. Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986a). Learning internal representations by error propagation. In D. E. Rumelhart, and J. L. McClelland (Eds.), Parallel distributed processing: Explorations in the microstructure of cognition. Vol. 1: Foundations (pp. 318--362). Cambridge, MA: MIT Press.

53. Amari, S. I. and Cichocki, A.: Adaptive blind signal processing – Neural network approaches, Proceedings of the IEEE, 86 (1998), 2026–2048.

54. Haykin, S.: Neural Networks: A Comprehensive Foundation, Prentice-Hall, Upper Saddle River, New Jersey, 1994.

55. Hochet, B., "Multivalued MOS memory for variable-synapse neural networks," Electronics Letters, Volume 25, Issue 10, 11 May 1989, pp. 669 – 670.

56. Hochet, B., Peiris, V., Abdo, S., and Declercq, M.J., "Implementation of a learning Kohonen neuron based on a new multilevel storage technique," Solid-State Circuits, IEEE Journal of, Volume 26, Issue 3, Mar 1991, pp. 262 – 267.

57. M. Mahowald, "VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function," Ph.D. Thesis, Computation and Neural Systems, California Institute of Technology, 1992.

58. A. Mortara, E.A. Vittoz, "A Communication Architecture Tailored for Analog VLSI Artificial Neural Networks: Intrinsic Performance and Limitations," IEEE Trans. Neural Networks, vol. 5, no. 3, May 1994, pp. 459 - 466.

59. E. Culurciello, R. Etienne-Cummings, K. A. Boahen, "A biomorphic digital image sensor," IEEE Journal of Solid-State Circuits, vol. 38, no. 2, February 2003, pp. 281–294.

60. M. Jabri and B. Flower, “Weight perturbation: an optimal architecture and learning technique for analog VLSI feedforward and recurrent multilayer networks,” IEEE Transactions on Neural Networks, Volume 3, Issue 1, January 1992, pp. 154 – 157.

61. G. Cauwenberghs, "A fast stochastic error-descent algorithm for supervised learning and optimisation," Advances in Neural Information Processing Systems, S.J. Hanson, J.D. Cowan and C.L. Giles Editors, Morgan Kaufmann Publishers, 5, 1993, pp. 244-251.

62. J. Alspector, R. Meir, B. Yuhas, and A. Jayakumar, "A parallel gradient descent method for learning in analog VLSI neural networks," Advances in Neural Information Processing Systems, S.J. Hanson, J.D. Cowan and C.L. Giles Editors, Morgan Kaufmann Publishers, 5, 1993, 836-844.

63. B.F. Flower and M.A. Jabri, "Summed Weight Neuron Perturbation: an O(N) improvement over Weight Perturbation. Morgan Kaufmann Publishers, NIPS, 5, 1993, pp. 212-219.

64. D. Buss, “Technology in the Internet age,” in ISSCC 2002 Dig. Tech. Papers, pp. 18–21.

65. D. Buss et al., “SOC CMOS technology for personal internet products,” IEEE Transactions on Electron Devices, vol. 50, no. 3, March 2003, pp. 546-556.

66. P. van Zeijl et al., “A Bluetooth radio in 0.18um CMOS,” IEEE Journal of Solid-State Circuits, vol. 37, no. 12, Dec. 2002, pp. 1679-1687.

67. Samavedam, A., Sadate, A., Mayaram, K., and Fiez, T.S., “A scalable substrate noise coupling model for design of mixed-signal IC’s,” IEEE Journal of Solid-State Circuits, vol. 35, no. 6, June 2000, pp. 895-904.

68. Yang, M.T. et al., "Characterization and model of on-chip flicker noise with deep Nwell (DNW) isolation for 130nm and beyond SOC," Proceedings of the 2005 International Conference on Microelectronic Test Structures, 4-7 April 2005, pp. 125-129.

69. Riter, S. and Boatright, P., "Design considerations for a pulse position modulation underwater acoustic communications system," OCEANS, Vol. 2, Sept. 1970, pp. 139 – 141.

70. Riter, S., Boatright, P., and Shay, M., "Pulse position modulation acoustic communications," IEEE Transactions on Audio and Electroacoustics, Vol. 19, no. 2, June 1971, pp. 166 – 173.

71. COOKE, D., JELONEK, Z., OXFORD, A.J., and FITCH, E., 'Pulse communication', J. IEE, 94, Part IIIA, 1947, pp. 83-105

72. FITCH, E., 'The spectrum of modulated pulses', J. IEE, 94, Part IIIA, 1947, pp. 556-564

73. JELONEK, Z., 'Noise problems in pulse communication', J. IEE, 94, Part IIIA, 1947, pp. 533-545

74. LEVY, M.M., 'Some theoretical and practical considerations of pulse modulation', J. IEE, 94, Part IIIA, 1947, pp. 565-572

75. SCHROCKS, C.B., 'Proposal for a hub controlled cable television system using optical fiber', IEEE Trans. on cable television, vol. CATV-4, no. 2, 1979, pp. 70-77.

76. BERRY, M.C., 'Pulse width modulation for optical fibre transmission,' PhD Thesis, Nottingham University, England, 1983.

77. BERRY, M.C., and ARNOLD, J.M., 'Pulse width modulation for optical fibre transmission of video'. IEE Int. Conf. on the Impact of VLSI Technology on Communication Systems, London, 1983.

78. SUH, S.Y., 'Pulse width modulation for analog fiber-optic communications', IEEE Journal of Lightwave Technology, vol. 5, no. 1, Jan. 1987, pp. 102-112.

79. WILSON, B., and GHASSEMLOOY, Z., 'Optical pulse width modulation for electrically isolated analogue transmission', J. Phys. (E), 1985, 18, pp. 954-958.

80. WILSON, B., and GHASSEMLOOY, Z., 'Optical PWM data link for high quality analogue and video signals', J. Phys. (E), 1987, 20, 7, pp. 841-845.

81. WILSON, B., and GHASSEMLOOY, Z., 'Optical fibre transmission of multiplexed video signals using PWM, Int. J. Optoelectron., 1989, 4, pp. 3-17.

82. HEATLEY, D.J.T., 'Video transmission in optical fibre local networks using pulse time modulation'. ECOC 83 - 9th European Conference on Optical Communication, Geneva, September 1983, pp. 343-346.

83. OKAZAKI, A., ‘Still picture transmission by pulse interval modulation’, IEEE Trans., 1979, CATV4 pp. 17-22.

84. HEATLEY, D.J.T., ‘Unrepeatered video transmission using pulse frequency modulation over 100 km of monomode optical fibre’, Electron. Lett., 1982, 18, pp. 369-371.

85. HEATLEY, D.J.T., and HODGKINSON, T.G., ‘Video transmission over cabled monomode fibre at 1.5 pm using PFM with 2-PSK heterodyne detection’, Electron. Lett., 1984,20, pp, 110-112.

86. HEKER, S.F., HERSKOWITZ, G.J., GREBEL, H., and WICHANSKY, H., ‘Video transmission in optical fiber communication systems using pulse frequency modulation’, IEEE Trans. Commun., 1988,36, (2), pp. 191-194.

87. KANADA, T., HAKODA, K., and YONEDA, E., ‘SNR fluctuation and non-linear distortion in PFM optical NTSC video transmission systems’, IEEE Trans. on Communications, 1982, 30, (8), pp. 1868-1875.

88. LU, C., ‘Optical transmission of wideband video signals using SWFM (PhD Thesis, University of Manchester Institute of Science and Technology, Manchester, England, 1990).

89. POPHILLAT, L., ‘Video transmission using a 1.3 pm LED and monomode fiter’, 10th European Conf. on Optical Communications, Stuttgart, W. Germany, 1984, pp. 238-239.

90. SATO, K., AOYGAI, S., and KITAMI, T., ‘Fiber optic video transmission employing square wave frequency modulation’, IEEE Trans., 1985iC6M-33,’(5), pp. 417-423.

91. WILSON, B., GHASSEMLOOY, Z., DARWAZEH, I., LU, C., and CHAN. D., ‘Optical sauarewave frequency modulation for wideband instrumentation ahd video signals’. IEE Colloquium on Analogue Optical Communications’, London, 1989, Digest 1989/165, Paper 9.

92. WILSON, B., GHASSEMLOOY, Z., and LU, C., ‘Optical fibre transmission of high-definition television signals using squarewave frequency modulation’. Third Bangor Symposium on Communications, University of Wales, Bangor, May 1991, pp. 258-262.

93. WILSON, B., GHASSEMLOOY, Z., and LU, C., ‘Squarewave FM optical fibre transmission for high definition television signals’ (Fibre Optics 90, 1990, London), Proc. Int. Soc. Optical Eng., 1990, 1314, pp. 90-97.

94. OKAZAKI, A., ‘Pulse interval modulation applicable to narrowband transmission’, IEEE Trans., 1978, CATV-3, pp. 155-164.

95. SATO, M., MURATA, M., and NAMEKAWA, T., ‘Pulse interval and width modulation for video transmission’, IEEE Trans., 1978, CATV-3, (4), pp. 166-173.

96. SATO, M., MURATA, M., and NAMEKAWA, T., ‘A new optical communication system using the Dulse interval and width modulated code’, IEEE-Trans, 1979, CATV-4, (l), pp. 1-9.

97. WILSON, B., GHASSEMLOOY, Z., and CHEUNG, J.C.S., ‘Spectral predictions for pulse interval and width modulation’, Electron.Lett., 1991, 27, (7), pp. 580-581.

98. Wilson, B. and Ghassemlooy, Z., "Pulse time modulation techniques for analogue optical fibre transmission," IEE Colloquium on Analogue Optical Communications, 18 Dec 1989, pp. 7/1-7/4.

99. Wilson, B., Ghassemlooy, Z., and Cheung, J.C.S., "Optical pulse interval and width modulation for analogue fibre communications,” IEE Proceedings Journal, Vol. 139, no. 6, Dec. 1992, pp. 376-382.

100. Ghassemlooy, Z. and Wilson, B., "Optical compound pulse time modulation for analogue fibre transmission," Proceedings of IEEE Singapore International Conference on Networks, Vol. 2, 6-11 Sept. 1993, pp. 630-634.

101. Wilson, B. and Ghassemlooy, Z., "Pulse time modulation techniques for optical communications: a review," IEE Proceedings Journal, Vol. 140, no. 6, Dec. 1993, pp. 347-357.

102. Lu, C., Wilson, B., and Ghassemlooy, Z., "Pulse time modulation techniques for low cost analog signal transmission systems," Proceedings of IEEE Singapore International Conference on Networks, vol. 2, 6-11 Sept. 1993, pp. 635-638.

103. Cowen, S., "Fiber Optic Video Transmission System Employing Pulse Frequency Modulation," OCEANS, vol. 11, Sep 1979, pp. 253-259.

104. DAS, J., and SHARMA, P.D.: ‘Pulse interval modulation’, Electron. Lett., 1967, 3, pp. 288-289.

105. Haykin, S., Communication Systems, 3rd ed., New York: Wiley, 1994.

106. R. Gabel and R. Roberts, Signals and Linear Systems, 3rd ed., New York: Wiley, 1987.

107. BLACK, H.S., Modulation theory, Princeton, N.J.: Van Nostrand, 1953.

108. W. Hayt and J. Kemmerly, Engineering Circuit Analysis, 5th ed., New York: McGraw-Hill, 1993.

109. E. Sanchez-Sinencio and A. Andreou, Low-Voltage/Low-Power Integrated Circuits and Systems: Low-Voltage Mixed-Signal Circuits, 1st ed., New York: Wiley, 1999.

110. 110. B. Maundy and P. Aronhime, “Useful multipliers for low-voltage applications,” Proc. IEEE Int. Symp. Circuits and Systems, vol. 1, May 2002, pp. 737–740.

111. B. Maundy and M. Maini, "A Comparison of Three Multipliers Based on the Vgs^2 Technique for Low-Voltage Applications," IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, vol. 50, no. 7, July 2003, pp. 937-940.

112. Y.P. Tsividis, Operation and Modeling of the MOS Transistor, New York: McGraw-Hill, 1987.

113. R.S. Muller and T.I. Kamins, Device Electronics for Integrated Circuits, New York: Wiley, 1986.

114. K. Bult and H. Wallinga, “A class of analog CMOS circuits based on the square-law characteristics of an MOS transistor in saturation,” IEEE J. Solid-State Circuits, vol. SC-22, no. 3, June 1987, pp. 357–365.

115. A. Stolmever. “A twin-well CMOS process using high energy ion implantation," IEEE Trans. Electron Devices, vol. ED-33, no. 4, Apr. 1986, pp. 450-457.

116. Mentor Graphics, Designing ASICs with the ADK Design Kit and Mentor Graphics Tools, Version 2.0, Dec. 2001.

117. Mentor Graphics, Cell Builder, Version 6.1.3, Oct. 2001.

118. TSMC 0.35um Process, http://www.mosis.org/products/fab/vendors/tsmc/tsmc035/, Feb. 10th, 2006.

119. MOSIS, Parametric Test Results, http://www.mosis.org/cgi-bin/cgiwrap/umosis/swp/params/tsmc-035/t4be_mm_epi-params.txt, Feb. 10th, 2006.

120. Lee, T.H. and Bulzacchelli, J.F., “A 155-MHz clock recovery delay- and phase-locked loop,” IEEE Journal of Solid-State Circuits, vol. 27, no. 12, Dec. 1992, pp. 1736-1746.

121. Hennessy, John L. and Patterson, David A., Computer Architecture: A Quantitative Approach, San Francisco, CA: Morgan Kaufman Publishers, 1996.

122. Maxim, “A Simple ADC Comparison Matrix,” Application Note 2094, Jun. 2nd, 2003.

123. Allen, Phillip E. and Holberg, Douglas R., CMOS Analog Circuit Design, 2nd ed., New York: Oxford University Press, Jan. 2002.

124. Yasumoto, M., Enomoto, T., Watanabe, K., and Ishihara, T., "Single-Chip Adaptive Transversal Filter IC Employing Switched Capacitor Technology," IEEE Journal on Selected Areas in Communications, vol. 2, no. 2, March 1984, pp.324-333.

125. Enomoto, T. and Yasumoto, M., "Integrated MOS four-quadrant analog multiplier using switched capacitor technology for analog signal processor ICs," Solid-State Circuits, IEEE Journal of, vol. 20, no. 4, Aug. 1985, pp. 852-859.

126. Grech, I., Micallef, J., and Vladimirova, T., "±0.9 V switched-capacitor CMOS multiplier with rail-to-rail input," Electronics Letters, vol. 35, no. 20, Sept. 1999, pp. 1688-1689.

127. Temes, G.C. and Tsividis, Y., "The special section on switched-capacitor circuits," Proceedings of the IEEE, vol. 71, no. 8, Aug. 1983, pp. 915-916.

128. Gregorian, R., Martin, K.W., and Temes, G.C., "Switched-capacitor circuit design," Proceedings of the IEEE, vol. 71, no. 8, Aug. 1983, pp. 941-966.

129. Tsividis, Y., "Principles of operation and analysis of switched-capacitor circuits," Proceedings of the IEEE, vol. 71, no. 8, Aug. 1983, pp. 926-940.

130. Allstot, D.J. and Black, W.C., Jr., "Technological design considerations for monolithic MOS switched-capacitor filtering systems," Proceedings of the IEEE, vol. 71, no. 8, Aug. 1983, pp. 967-986.

131. National Semiconductor, “A Basic Introduction to Filters - Active, Passive, and Switched-Capacitor,” Application Note 779, Apr. 1991.

Appendix A. MOSIS Parametric Test Results

```
* Source: http://www.mosis.org/cgi-bin/cgiwrap/umosis/swp/params/tsmc-035/t4be_mm_epi-
params.txt, Feb. 10th, 2006.

                          MOSIS PARAMETRIC TEST RESULTS

          RUN: T4BE (MM_EPI)                                  VENDOR: TSMC
   TECHNOLOGY: SCN035                                  FEATURE SIZE: 0.35 microns


INTRODUCTION: This report contains the lot average results obtained by MOSIS
              from measurements of MOSIS test structures on each wafer of
              this fabrication lot. SPICE parameters obtained from similar
              measurements on a selected wafer are also attached.

COMMENTS: TSMC 035


TRANSISTOR PARAMETERS     W/L      N-CHANNEL P-CHANNEL  UNITS

 MINIMUM                  0.6/0.4
  Vth                                   0.55     -0.76  volts

 SHORT                    20.0/0.4
  Idss                                513      -220     uA/um
  Vth                                   0.59     -0.75  volts
  Vpt                                   9.1      -9.7   volts

 WIDE                     20.0/0.4
  Ids0                                < 2.5     < 2.5   pA/um

 LARGE                    50/50
  Vth                                   0.52     -0.75  volts
  Vjbkd                                 8.7      -8.5   volts
  Ijlk                                <50.0     <50.0   pA
  Gamma                                 0.60      0.37  V^0.5

 K' (Uo*Cox/2)                         89.1     -30.8   uA/V^2
 Low-field Mobility                   402.53    139.15  cm^2/V*s

COMMENTS: Poly bias varies with design technology. To account for mask
           bias use the appropriate value for the parameter XL in your
           SPICE model card.
                       Design Technology                  XL (um)  XW (um)
                       -----------------                  -------  ------
                       SCMOS_SUBM (lambda=0.20)           -0.05     0.15
                                     thick oxide          -0.10     0.15
                       SCMOS      (lambda=0.25)           -0.15     0.15
                                     thick oxide          -0.25     0.15


FOX TRANSISTORS           GATE      N+ACTIVE  P+ACTIVE  UNITS
 Vth                      Poly        >10.0    <-10.0   volts



PROCESS PARAMETERS     N+     P+    POLY  POLY2  POLY2_ME   M1    M2   UNITS
 Sheet Resistance      78.9  151.7  8.5   48.6      48.6   0.07  0.07  ohms/sq
 Contact Resistance    62.5         6.8                                ohms
 Gate Oxide Thickness  78                                              angstrom
PROCESS PARAMETERS             M3        M4    N_W      N\PLY     UNITS
```

```
 Sheet Resistance             0.07      0.04   1006      1050     ohms/sq
 Contact Resistance           2.06      2.99                      ohms

COMMENTS: N\POLY is N-well under polysilicon.


CAPACITANCE PARAMETERS  N+    P+   POLY  POLY2  M1  M2   M3   M4  N_W   UNITS
 Area (substrate)      914  1401   101                               99  aF/um^2
 Area (N+active)                  4440              17   12   10         aF/um^2
 Area (P+active)                  4478                                   aF/um^2
 Area (poly)                             868    49  15    9    6         aF/um^2
 Area (poly2)                                   47                       aF/um^2
 Area (metal1)                                      36   14    8         aF/um^2
 Area (metal2)                                           37   13         aF/um^2
 Area (metal3)                                                36         aF/um^2
 Fringe (substrate)    251   295                                         aF/um
 Fringe (poly)                                  67  38   29   23         aF/um
 Fringe (metal1)                                    51   34   27         aF/um
 Fringe (metal2)                                              37         aF/um
 Fringe (metal3)                                              57         aF/um
 Overlap (N+active)                340                                   aF/um
 Overlap (P+active)                383                                   aF/um



CIRCUIT PARAMETERS                            UNITS
 Inverters                     K
  Vinv                        1.0       1.21  volts
  Vinv                        1.5       1.35  volts
  Vol (100 uA)                2.0       0.23  volts
  Voh (100 uA)                2.0       2.88  volts
  Vinv                        2.0       1.45  volts
  Gain                        2.0     -17.91
 Ring Oscillator Freq.
  DIV256 (31-stg,3.3V)              175.82  MHz
  D256_THK  (31-stg,5.0V)           112.34  MHz
 Ring Oscillator Power
  DIV256 (31-stg,3.3V)                0.15  uW/MHz/gate
  D256_THK  (31-stg,5.0V)             0.30  uW/MHz/gate

COMMENTS: SUBMICRON




T4BE SPICE BSIM3 VERSION 3.1 PARAMETERS

SPICE 3f5 Level 8, Star-HSPICE Level 49, UTMOST Level 8

* DATE: Dec 17/04
* LOT: T4BE                  WAF: 1005
* Temperature_parameters=Default
.MODEL CMOSN NMOS (                                LEVEL   = 49
+VERSION = 3.1            TNOM    = 27             TOX     = 7.8E-9
+XJ      = 1E-7           NCH     = 2.2E17         VTH0    = 0.4867759
+K1      = 0.5982003      K2      = 7.110775E-3    K3      = 74.853704
+K3B     = -10            W0      = 5E-5           NLX     = 2.522198E-7
+DVT0W   = 0              DVT1W   = 0              DVT2W   = 0
+DVT0    = 2.8489298      DVT1    = 0.8382014      DVT2    = -0.2705174
+U0      = 364.7816729    UA      = -7.07762E-10   UB      = 2.214939E-18
+UC      = 3.217876E-11   VSAT    = 1.427282E5     A0      = 1.0943978
+AGS     = 0.1518849      B0      = 9.486928E-7    B1      = 5E-6
+KETA    = 2.703951E-3    A1      = 1.331977E-4    A2      = 0.5082922
+RDSW    = 1.03395E3      PRWG    = -0.0999764     PRWB    = -0.1019712
+WR      = 1              WINT    = 1.517735E-7    LINT    = 0
+XL      = -5E-8          XW      = 1.5E-7         DWG     = -3.822451E-9
+DWB     = 3.625123E-9    VOFF    = -0.0834415     NFACTOR = 1.4572053
+CIT     = 0              CDSC    = 2.4E-4         CDSCD   = 0
+CDSCB   = 0              ETA0    = 1              ETAB    = 0.0333683
+DSUB    = 0.8054936      PCLM    = 1.4582292      PDIBLC1 = 1.807793E-3
+PDIBLC2 = 1.210914E-4    PDIBLCB = 0.0219649      DROUT   = 3.881351E-4
+PSCBE1  = 7.237622E8     PSCBE2  = 1E-3           PVAG    = 0
+DELTA   = 0.01           RSH     = 78.9           MOBMOD  = 1
+PRT     = 0              UTE     = -1.5           KT1     = -0.11
+KT1L    = 0              KT2     = 0.022          UA1     = 4.31E-9
+UB1     = -7.61E-18      UC1     = -5.6E-11       AT      = 3.3E4
+WL      = 0              WLN     = 1              WW      = 0
+WWN     = 1              WWL     = 0              LL      = 0
+LLN     = 1              LW      = 0              LWN     = 1
+LWL     = 0              CAPMOD  = 2              XPART   = 0.5
+CGDO    = 3.4E-10        CGSO    = 3.4E-10        CGBO    = 1E-12
+CJ      = 9.085053E-4    PB      = 0.8            MJ      = 0.3519913
+CJSW    = 2.378884E-10   PBSW    = 0.8            MJSW    = 0.1980245
+CJSWG   = 1.82E-10       PBSWG   = 0.8            MJSWG   = 0.1980245
+CF      = 0              PVTH0   = -0.0232737     PRDSW   = -96.6778453
+PK2     = 4.200437E-3    WKETA   = -9.87476E-4    LKETA   = -1.128415E-4    )
*
.MODEL CMOSP PMOS (                                LEVEL   = 49
+VERSION = 3.1            TNOM    = 27             TOX     = 7.8E-9
+XJ      = 1E-7           NCH     = 8.52E16        VTH0    = -0.7127409
+K1      = 0.426131       K2      = -8.52838E-3    K3      = 53.6180914
+K3B     = -4.1856702     W0      = 4.117358E-6    NLX     = 2.421492E-7
+DVT0W   = 0              DVT1W   = 0              DVT2W   = 0
+DVT0    = 1.697126       DVT1    = 0.5602561      DVT2    = 1.795544E-3
+U0      = 150.0058707    UA      = 1.051947E-10   UB      = 1.832285E-18
+UC      = -2.03335E-11   VSAT    = 1.149727E5     A0      = 1.1206664
+AGS     = 0.3393834      B0      = 2.49922E-6     B1      = 5E-6
+KETA    = -2.847654E-3   A1      = 0              A2      = 0.7697629
+RDSW    = 4E3            PRWG    = -0.12597       PRWB    = 0.1740295
+WR      = 1              WINT    = 1.54103E-7     LINT    = 0
+XL      = -5E-8          XW      = 1.5E-7         DWG     = -1.498245E-8
+DWB     = 7.89723E-9     VOFF    = -0.135593      NFACTOR = 2
+CIT     = 0              CDSC    = 2.4E-4         CDSCD   = 0
+CDSCB   = 0              ETA0    = 0.0150933      ETAB    = 2.717128E-3
+DSUB    = 0.2430503      PCLM    = 4.39905        PDIBLC1 = 7.528966E-4
+PDIBLC2 = 3.163274E-3    PDIBLCB = -1E-3          DROUT   = 8.227276E-3
+PSCBE1  = 7.936256E10    PSCBE2  = 5E-10          PVAG    = 3.1637761
+DELTA   = 0.01           RSH     = 151.7          MOBMOD  = 1
+PRT     = 0              UTE     = -1.5           KT1     = -0.11
+KT1L    = 0              KT2     = 0.022          UA1     = 4.31E-9
+UB1     = -7.61E-18      UC1     = -5.6E-11       AT      = 3.3E4
+WL      = 0              WLN     = 1              WW      = 0
+WWN     = 1              WWL     = 0              LL      = 0
+LLN     = 1              LW      = 0              LWN     = 1
+LWL     = 0              CAPMOD  = 2              XPART   = 0.5
+CGDO    = 3.83E-10       CGSO    = 3.83E-10       CGBO    = 1E-12
+CJ      = 1.404522E-3    PB      = 0.99           MJ      = 0.5637342
+CJSW    = 2.976181E-10   PBSW    = 0.8386314      MJSW    = 0.350651
+CJSWG   = 4.42E-11       PBSWG   = 0.8386314      MJSWG   = 0.350651
+CF      = 0              PVTH0   = 7.323739E-3    PRDSW   = 57.2472155
+PK2     = 1.982984E-3    WKETA   = -3.979881E-4   LKETA   = -0.0121896      )
*
```

Appendix B. BSIM3v3.1 SPICE Models Used for ACCUSIM Simulations

```
* In Mentor ASIC Design Kit (ADK), Accusim uses LEVEL 53 for BSIM3v3.1 SPICE Modeling.
* So, LEVEL was set to 53 instead of 49 for HSPICE.



.MODEL n NMOS (                                LEVEL   = 53
+VERSION = 3.1            TNOM    = 27            TOX     = 7.7E-9
+XJ      = 1E-7           NCH     = 2.2E17        VTH0    = 0.4737706
+K1      = 0.5824799      K2      = 8.11778E-3    K3      = 96.2727323
+K3B     = -6.1947761     W0      = 2.700191E-5   NLX     = 2.083266E-7
+DVT0W   = 0              DVT1W   = 0             DVT2W   = 0
+DVT0    = 4.3980137      DVT1    = 0.7171277     DVT2    = -0.1080294
+U0      = 362.8973445    UA      = -7.42132E-10  UB      = 2.200466E-18
+UC      = 3.827538E-11   VSAT    = 1.452933E5    A0      = 1.1270999
+AGS     = 0.1655255      B0      = 1.060433E-6   B1      = 5E-6
+KETA    = 2.011502E-3    A1      = 0             A2      = 0.4936368
+RDSW    = 855.0105543    PRWG    = -0.0459015    PRWB    = -0.0966899
+WR      = 1              WINT    = 1.486341E-7   LINT    = 5.292541E-10
+XL      = -5E-8          XW      = 1.5E-7        DWG     = -4.068542E-9
+DWB     = 8.2373E-9      VOFF    = -0.0796631    NFACTOR = 1.2068796
+CIT     = 0              CDSC    = 2.4E-4        CDSCD   = 0
+CDSCB   = 0              ETA0    = 0.7492832     ETAB    = -0.0390417
+DSUB    = 0.78159        PCLM    = 1.490984      PDIBLC1 = 1.944604E-3
+PDIBLC2 = 5.987715E-7    PDIBLCB = 0.1           DROUT   = 0
+PSCBE1  = 7.310861E8     PSCBE2  = 9.494637E-4   PVAG    = 0
+DELTA   = 0.01           RSH     = 79.8          MOBMOD  = 1
+PRT     = 0              UTE     = -1.5          KT1     = -0.11
+KT1L    = 0              KT2     = 0.022         UA1     = 4.31E-9
+UB1     = -7.61E-18      UC1     = -5.6E-11      AT      = 3.3E4
+WL      = 0              WLN     = 1             WW      = 0
+WWN     = 1              WWL     = 0             LL      = 0
+LLN     = 1              LW      = 0             LWN     = 1
+LWL     = 0              CAPMOD  = 2             XPART   = 0.5
+CGDO    = 2.79E-10       CGSO    = 2.79E-10      CGBO    = 1E-12
+CJ      = 8.918562E-4    PB      = 0.8           MJ      = 0.3534487
+CJSW    = 3.57525E-10    PBSW    = 0.8146587     MJSW    = 0.1331765
+CJSWG   = 1.82E-10       PBSWG   = 0.8146587     MJSWG   = 0.1331765
+CF      = 0              PVTH0   = -0.0199259    PRDSW   = -87.5466049
+PK2     = 2.754154E-3    WKETA   = -7.698947E-4  LKETA   = -2.706215E-3    )



.MODEL p PMOS (                                LEVEL   = 53
+VERSION = 3.1            TNOM    = 27            TOX     = 7.7E-9
+XJ      = 1E-7           NCH     = 8.52E16       VTH0    = -0.7482928
+K1      = 0.4088906      K2      = -8.575971E-3  K3      = 64.9719334
+K3B     = -5             W0      = 5.652246E-6   NLX     = 2.411327E-7
+DVT0W   = 0              DVT1W   = 0             DVT2W   = 0
+DVT0    = 2.4385401      DVT1    = 0.6534475     DVT2    = 0.0336004
+U0      = 146.6363388    UA      = 1E-10         UB      = 1.554136E-18
+UC      = -2.50914E-11   VSAT    = 1.158448E5    A0      = 0.9834203
+AGS     = 0.3498974      B0      = 3.014933E-6   B1      = 5E-6
+KETA    = -5.567367E-3   A1      = 0             A2      = 0.6767726
+RDSW    = 4E3            PRWG    = -0.1174124    PRWB    = 0.1956899
+WR      = 1              WINT    = 1.481075E-7   LINT    = 0
+XL      = -5E-8          XW      = 1.5E-7        DWG     = -1.237406E-8
+DWB     = 1.314322E-8    VOFF    = -0.1341843    NFACTOR = 2
+CIT     = 0            CDSC     = 2.4E-4          CDSCD   = 0
+CDSCB   = 0            ETA0     = 0.0247927       ETAB    = -5.470312E-3
+DSUB    = 0.4066258    PCLM     = 4.3395303       PDIBLC1 = 3.712474E-3
+PDIBLC2 = 2.843199E-3  PDIBLCB  = -1E-3           DROUT   = 0.0493039
+PSCBE1  = 7.972605E10  PSCBE2   = 5.002845E-10    PVAG    = 2.4985005
+DELTA   = 0.01         RSH      = 154.7           MOBMOD  = 1
+PRT     = 0            UTE      = -1.5            KT1     = -0.11
+KT1L    = 0            KT2      = 0.022           UA1     = 4.31E-9
+UB1     = -7.61E-18    UC1      = -5.6E-11        AT      = 3.3E4
+WL      = 0            WLN      = 1               WW      = 0
+WWN     = 1            WWL      = 0               LL      = 0
+LLN     = 1            LW       = 0               LWN     = 1
+LWL     = 0            CAPMOD   = 2               XPART   = 0.5
+CGDO    = 2.75E-10     CGSO     = 2.75E-10        CGBO    = 1E-12
+CJ      = 1.429787E-3  PB       = 0.99            MJ      = 0.5495301
+CJSW    = 3.794122E-10 PBSW     = 0.99            MJSW    = 0.3012354
+CJSWG   = 4.42E-11     PBSWG    = 0.99            MJSWG   = 0.3012354
+CF      = 0            PVTH0    = 3.478791E-3     PRDSW   = 33.4845306
+PK2     = 1.529109E-3  WKETA    = 2.584294E-3     LKETA   = -1.775326E-3    )
```