Variable Supply Voltage for Low Power DSP

by

Vadim Gutnik

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1996

Author .....
Department of Electrical Engineering and Computer Science
May 24, 1996

Certified by .....
Anantha Chandrakasan
Associate Professor
Thesis Supervisor

Accepted by .....
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Students

Variable Supply Voltage for Low Power DSP

by

Vadim Gutnik

Submitted to the Department of Electrical Engineering and Computer Science
on May 24, 1996, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering

Abstract

The use of dynamically adjustable power supplies as a method to lower power dissipation in DSP is analyzed. Power can be cut substantially without sacrificing performance in applications where the computational workload changes with time. If latency can be tolerated, buffering data and averaging processing rate can yield further power reduction. Continuous variation of the supply voltage can be approximated by very crude quantization and dithering. A chip has been fabricated and tested to verify the implementation of these ideas and to demonstrate the general control framework developed in this thesis.

Thesis Supervisor: Anantha Chandrakasan
Title: Associate Professor

Acknowledgments

I would like to thank my thesis advisor, Professor Anantha Chandrakasan who not only came up with the initial idea for the project but also provided the encouragement to finish it. Thanks also to Professors Charles Sodini and Martin Schlecht for offering constructive criticism.

Thanks goes to my research group as well — to Jim Goodman for finding my mistakes and to Tom Simon for finding the ones I was going to make. To Raj Amirtharajah for telling me not to worry about deadlines, and to Duke Xanthopoulos and Abram Dancy for listening to me rant about variable voltage supplies. To Tom Barber who showed by example what it takes to finish a thesis. To Carlin Vieri and especially Mike Bolotski in Tom Knight's group for bandaging Cadence when it fell apart, and to everyone at SIPB who helped me when my computer would act up.

And of course, thanks to Mom and Dad, for reasons too numerous to recount.

Contents

List of Figures

Chapter 1

Overview of Low Power ICs

In the last several years power consumption of integrated circuits has become an increasingly important design constraint. One reason has been the demand for longer battery life from portable electronics applications like laptop computers and mobile phones; control of power consumption has proved to be a more significant factor than battery technology for increasing run-time. Another factor has been the the increasing power density of microprocessors; the package and circuit board have to dissipate the $\approx 30W$ of heat that a Dec Alpha 21164 or Pentium Pro processor generates.

1.1 Low power hierarchy

Power dissipation in CMOS circuits is proportional to the switched capacitance and the supply voltage squared; a reduction in either will lead to power savings. Not surprisingly, a wide variety of methods to lower power have been investigated. These can broadly classified by level of abstraction: technology, circuit design, architecture, and algorithm [5].

1.1.1 Technology

From the standpoint of system design the term "technology" includes everything from device doping levels to layout design rules to packaging. Some of the most

dramatic improvements in low power integrated circuits have come from technological advances. For example, static power dissipation was dramatically reduced when CMOS processes replaced NMOS processes in the 60's. In the 70's, ion implantation allowed better control of threshold voltages so the supply voltage (and therefore power) could be lowered [6].

One of the enduring trends in the semiconductor industry has been the scaling down of feature sizes on ICs. Since the invention of the integrated circuit in 1958, feature sizes have fallen by a factor of more than 50. Although this "technology scaling" was pursued primarily for economic motives the power savings substantial, since capacitance scales roughly with feature size. Feature scaling is expected to continue for the next 10 to 20 years, though not at the current rate of improvement. Silicon on insulator, or *SOI* technology is becoming more widespread as a way to lower parasitic capacitances and leakage power even further. As supply voltages drop the use of multi-V_t processes and back-gate biasing to optimize standby power consumption and gate delay will become more common [7, 8].

1.1.2 Circuit design

A variety of circuit approaches to minimizing power are available for specific applications. For example, sense amplifiers for highly capacitive lines allow small voltage swings, and therefore lower power, in critical sections of memories and similar modules. Transistor sizes can also be optimized — too large a device needlessly increases capacitance, while too small a transistor slows logic signals disproportionately.

Boolean logic minimization is also routinely employed to lower power. Unlike optimization for speed, where number of gate delays in the critical path is key, optimization for low power also factors in the total number of transistors and especially the switching activity on the internal nodes. Just as CAD tools are used to optimize for speed and area, optimization for power is becoming more automated.

1.1.3 Architecture

A large fraction of current research in low power is aimed at optimizing circuit architectures. Pipelining and parallelization are mainstay techniques in microprocessor design, but many other architecture decisions affect power as well. In the case of microprocessors, the size and access frequency of the instruction and data caches, the number of registers, clock distribution, the number of busses, and even the instruction set affects power performance.

Even more issues come up in DSP design. Number representation, for example, is the focus of some discussion. Is sign & magnitude or one-hot coding better than two's complement? Some of the other issues include the use of parallel *vs.* time-multiplexed resources, chain *vs.* tree arithmetic structures, synchronous *vs.* asynchronous design, *etc.* The tradeoff between power, speed, and area is explicit in architecture design.

1.1.4 Algorithm

The premise behind algorithmic optimization is so intuitive as to rarely be stated explicitly: simpler algorithms require smaller, lower power circuits. One of the most familiar and oldest examples of algorithmic optimization is the Fast Fourier transform. Direct computation of a length-N Discrete Time Fourier Transform takes N^2 operations; an FFT computes the same result in only $N \log_2 N$. For a modest 512 element transform, that corresponds to 50-fold reduction in the number operations. A system that implements an FFT could tolerate longer gate delays and a lower voltage than one the computes the DTFT directly if the total processing time is fixed.

The biggest drawback to optimization at the algorithmic level is that the solutions are specific to very closely related problems; the FFT can be modified to compute a discrete cosine transform, for example, but contributes nothing to microcode execution or factoring integers. On the other hand, algorithm improvements are readily transferred to new technologies and circuit styles.

1.2 Dynamic optimization

The common thread through these techniques is that the optimization for low power is static; that is, whatever parameter is being optimized, be it supply voltage or circuit layout or state-assignment in an finite state machine, is optimized for low power at design time. In fact, in some applications optimizing the circuit *during operation* could save more power.

A handful of methods exist for optimizing circuit performance at run time; adaptive filters adjust the coefficients of their taps as data comes in, for example. And power down techniques strive to eliminate power consumption of idle blocks. However, in almost all cases the power supply is treated as a fixed voltage, and the clock as a fixed frequency.[1] In fact, the system clock and supply voltage can also be adjusted dynamically to lower overall system power. This optimization is the focus of this thesis.

1.3 Previous work on variable power supplies

A key previous work on variable power supplies involves compensating process and temperature variations by adjusting the power supply. Since threshold voltages cannot be controlled precisely, conventional ICs are designed to work under worst-case conditions. If actual circuit delays could be measured, the supply voltage could be lowered until the actual circuit delays match the maximum allowed delays. This idea can be carried further by using self-timed circuits where global clocks are eliminated completely and local handshaking establishes timing; both approaches are described below.

[1] Adiabatic logic families have AC rather than DC power supplies, but the power waveform is still prescribed at design time.

1.3.1 Dynamic process compensation

Several systems have been designed to compensate for process and temperature variations [9, 10, 1]. In each case the circuit delays are measured indirectly by means of a ring oscillator with a period matched to circuit delays. The ring oscillator is used as the VCO of a phase-locked loop; when the PLL settles the supply voltage for the oscillator is at exactly the point where circuit delays equal the clock period, so it can be used as a reference for the supply voltage for the chip. A simplified schematic is shown in Fig.1-1.

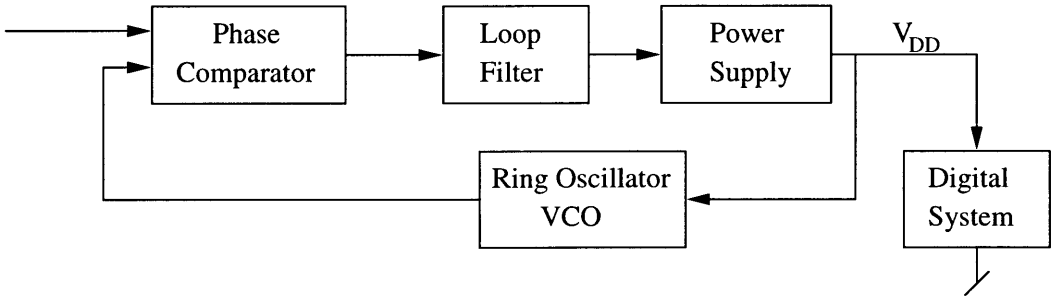

Figure 1-1: Supply voltage PLL schematic [1]

The power and area overhead is small; a single loop may be sufficient for the entire chip. Stability concerns limit the speed of the loop, but since temperature changes slowly, the bandwidth is adequate.

1.3.2 Self-timed circuits

Self-timed circuits can be used to generate timing signals based directly on circuit delays. Conventional circuits use a single clock to mediate transfer of data between all combinational blocks, as shown in Fig. 1-2.

Self-timed circuits can be used to eliminate these global clocks entirely. Instead of using a single clock, each logic block generates completion signals, and data transfer between blocks is accomplished by handshaking, as shown in Fig. 1-3 [4].

The completion signals depend on the actual latency in the combinational logic block. Thus, the block throughput depends not only on the number of gate delays, but also on the device parameters, the temperature, and even on the data being

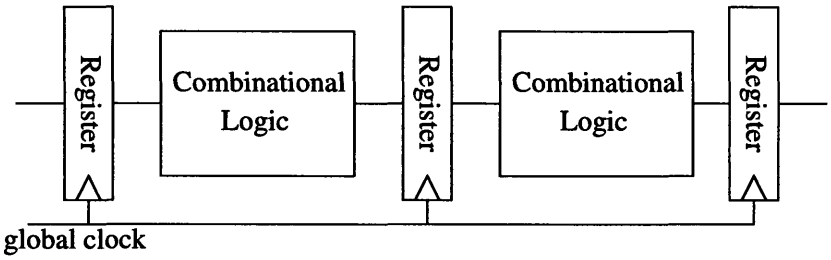

Figure 1-2: Global clocks

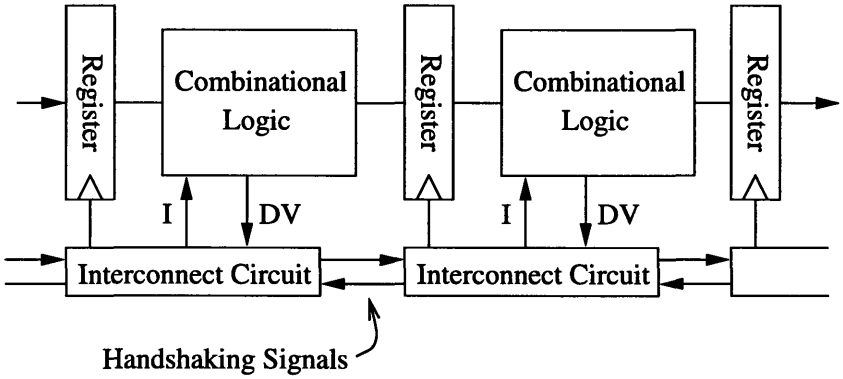

Figure 1-3: Local timing generation

processed at a bit level. This has been observed to give widely varying computation times for different data [4, 2].

Just as in the case of synchronous systems, the supply voltage can be lowered to save power until the throughput just meets system requirements. A small self-timed, variable voltage supply has been demonstrated by Nielssen *et. al.* [2]. The system, diagrammed in Fig. 1-4, consists of two FIFO buffers, a self-timed 16-bit ripple-carry

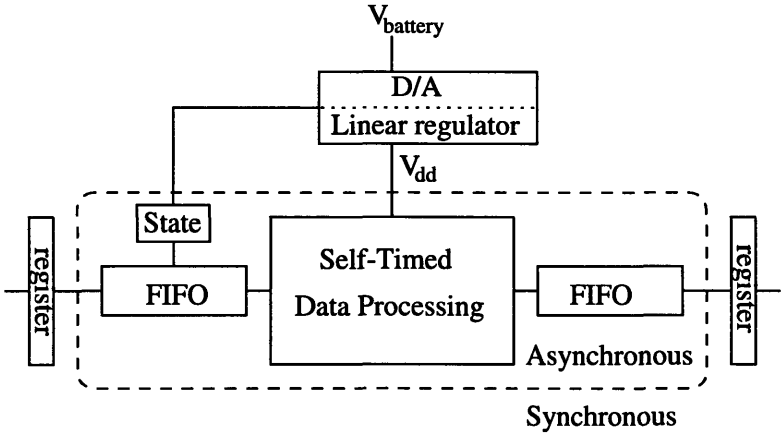

Figure 1-4: Self-timed system diagram [2]

adder and a D/A driving a power supply. The length of the ripple-carry propagation affects how long each addition takes. By monitoring the number of elements in the input queue the system automatically determines whether the processing is too fast or too slow and adjusts the supply voltage accordingly.

Of course, the entire scheme depends on generation of the completion signals. This and other self-timed systems have been designed with slight variations on Differential Cascode Voltage Switch Logic (DCVSL), a precharged-differential logic family similar to domino logic [3]. A generalized DCVSL gate that generates a completion signal is shown in Fig. 1-5. During precharge, the NMOS pulldown chain is disabled (**I** low), both logic outputs are low and **DV** is low. After **I** rises evaluation begins and **DV** is asserted as soon as one of the outputs rises.

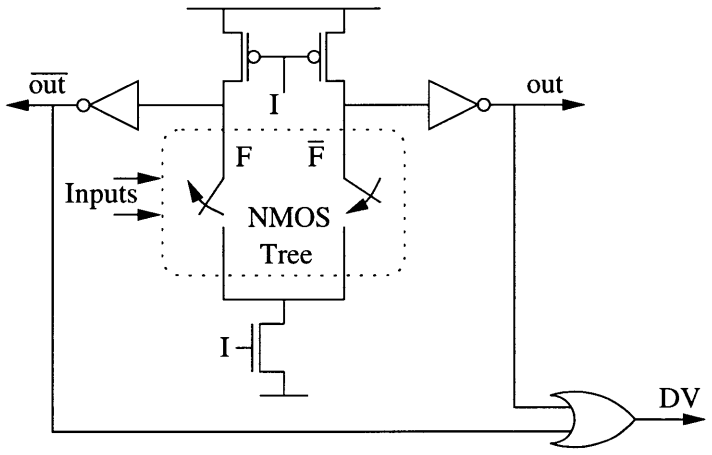

Figure 1-5: DCVSL gate [3, 4]

The generation of the completions signals is one weakness of this approach. First, the completion generation introduces extra gate delays to the signal path. Even worse from the standpoint of low-power is that the activity factor is 1 on the outputs of the gate. For comparison, a three-input static CMOS NAND gate has an activity factor of 7/64, and no power is dissipated when the outputs do not change.

1.4 Thesis focus

There is no way to exploit the bit-level data dependencies without resorting to a self-timed system. However, in many applications the processing requirements change at an algorithmic level, and these variations can be exploited by any synchronous system.

For example, an adaptive FIR filter has been proposed that could adapt the amount of processing based on the input noise: when the input noise level is relatively low, less processing needs to be done to the signal [11]. Fig 1-6 shows a simulated plot of filter order (*i.e.* the computation rate) changing with time as the noise on the input data changes.

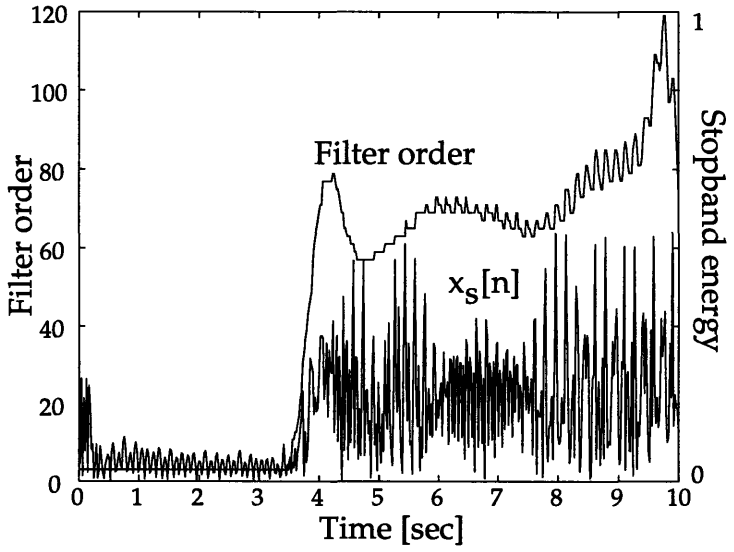

Figure 1-6: Variation in filter order as noise level changes

A conventional design of the filter would shut down sections or gate the clock when the noise level is low, saving power linearly with computation. If instead of shutting parts down the clock frequency and supply voltage were adjusted dynamically, the power savings would be much greater- slowing down by 25% saves a factor of 2!

Digital video is another application well suited to variable voltage. Since frames of video are highly correlated, most digital processing begins by comparing consecutive frames and processing only the differences. The amount of computation necessary

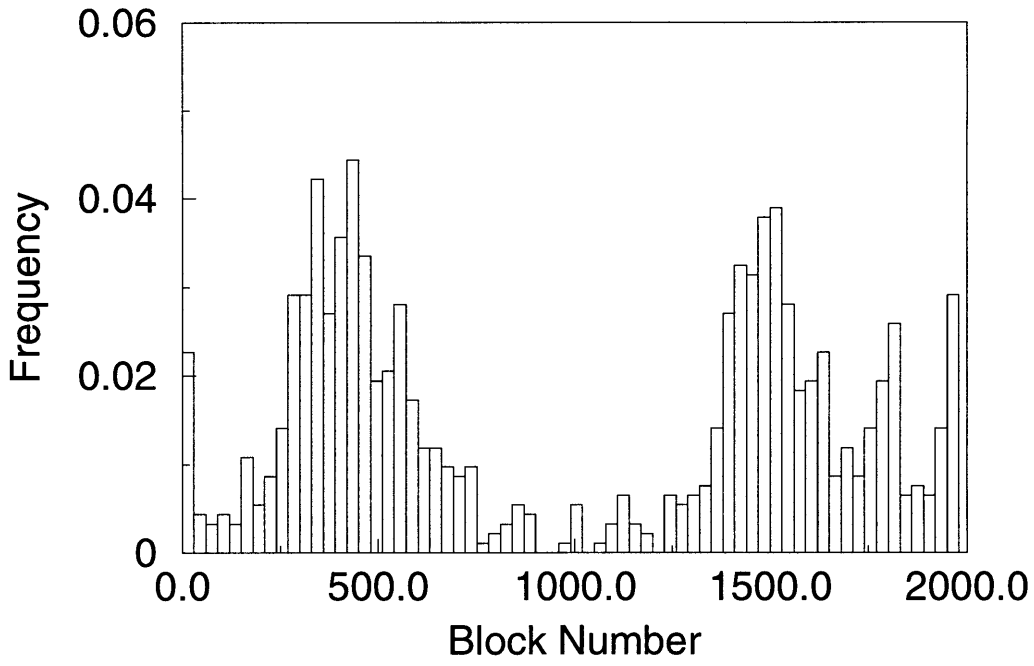

Figure 1-7: Histogram of the number of blocks processed per frame

to process the differences depends on how much the image changes from frame to frame. Typically each frame is divided into small blocks and only those that change more than a threshold are processed. Fig. 1-7 shows a histogram of the number of blocks that changed in sequence of MPEG2 video. The wide variation suggests that significant power savings are possible by dynamically adjusting supply voltage.

A synchronous system can exploit these algorithmic variations in workload to lower power. Like the self-timed variable-voltage system, some buffering would be needed on the input and output for synchronization, but a synchronous system could be designed without the completion generation signals, using static CMOS or any other logic family. Like the previous process-compensation scheme for a synchronous system, the clock would be generated with a PLL, but the loop can be optimized to allow faster voltage transitions when the workload changes. A block diagram of the system developed in this work is shown in Fig. 1-8.

The goal of this thesis is to analyze the power savings possible with a variable supply system and to determine how the implementation of such a system affects performance. In particular, previously unaddressed questions of buffer sizing and

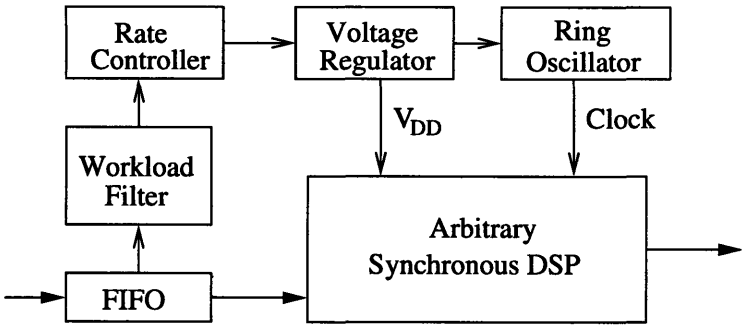

Figure 1-8: Synchronous, workload-dependent variable voltage system

power supply controller are treated. Although the emphasis will be on synchronous, rather than self-timed systems because of their lower capacitive overhead, most of the results could be adapted easily to either case.

Chapter 2

Variable Supply Fundamentals

Since the goal of a variable supply system is to minimize power, it is best to start by considering what sets the power dissipation of a digital system. If both the technology and architecture are fixed, the determining factor in power dissipation is timing: at higher voltages, the circuits operate faster but burn more power. The analysis begins by deriving the relationship of dissipated power to processing speed.

2.1 First order derivation

A convenient formalism is that digital signal processing deals with data as discrete *samples*. A sample may be as small as 12 bits in the case of audio processing or as large as a 1,000,000 bits for a full frame of video, but the data always comes in discrete chunks. In most cases the samples come at a fixed frequency; hence, the average power and computation for the processor can be related to the energy per sample and the computation performed therein.

In CMOS circuits most of the power goes to charging and discharging capacitances, so a starting approximation for the energy per sample is

$$E = nCV_{DD}^2 \tag{2.1}$$

where n is the number of clock cycles per sample time, C is the average switched

capacitance per clock cycle, and V_{DD} the supply voltage. The number of clock cycles per sample time is constrained by gate delay, since there has to be enough time for signals to propagate from one register to the next between clock cycles; the timing model is shown in Fig. 2-1.

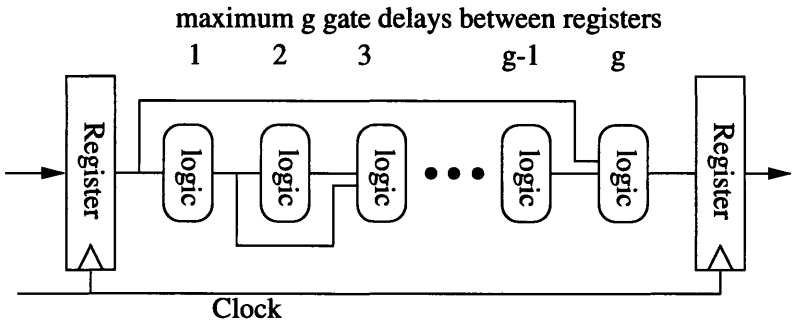

Figure 2-1: Timing constraint model

Defining g to be the number of gate delays between registers,[1] the maximum clock frequency is constrained to be $1/gT_d$, so to first order the maximum operating frequency is [12]:

$$f = 1/gT_d = \frac{\mu C_{ox}(W/L)(V_{DD} - V_t)^2}{g \cdot C_L V_{DD}} = \frac{\alpha(V_{DD} - V_t)^2}{V_{DD}} \tag{2.2}$$

For the purpose of this analysis, the device and circuit layout parameters are absorbed into α. The resulting equation depends only on the threshold voltage V_t and supply voltage V_{DD}. Equations 2.1 and 2.2 are typically stated for fixed-supply applications, where V_{DD} is a constant voltage; however, they remain true even if V_{DD} varies, provided that the variation is slow compared to gate delays. The combination of the two equations gives

$$\begin{aligned} E(f) &= nC\left[V_t + \frac{f}{2\alpha} + \sqrt{\frac{fV_t}{\alpha} + \left(\frac{f}{2\alpha}\right)^2}\right]^2 \\ &= nCV_0^2\left[\frac{V_t}{V_0} + \frac{f}{2f_{ref}} + \sqrt{\frac{f}{f_{ref}}\frac{V_t}{V_0} + \left(\frac{f}{2f_{ref}}\right)^2}\right]^2 \end{aligned} \tag{2.3}$$

[1]The setup and hold time requirements could be included as extra gate delays.

where $V_0 = f_{ref}/\alpha = (V_{ref} - V_t)^2/V_{ref}$ and V_{ref} is a reference supply voltage. Since the sample time is fixed, the clock frequency determines how many operations are performed per sample. If T_s is the sample time, the number of clock cycles is given by $n = fT_s$, and Eq. 2.3 can be rewritten as

$$\begin{aligned} E(f) &= CV_0{}^2 fT_s \left[\frac{V_t}{V_0} + \frac{f}{2f_{ref}} + \sqrt{\frac{f}{f_{ref}}\frac{V_t}{V_0} + \left(\frac{f}{2f_{ref}}\right)^2}\right]^2 \\ E(r) &= E_0 r \left[\frac{V_t}{V_0} + r/2 + \sqrt{r\frac{V_t}{V_0} + (r/2)^2}\right]^2 \end{aligned} \tag{2.4}$$

where E_0 is a scaling factor with units of energy and $r = f/f_{ref}$ is the *normalized sample processing rate*, or simply *rate*. Here f_{ref} is the frequency of operation at $V_{DD} = V_{ref}$, so if V_{ref} is the highest available voltage, f_{ref} is the maximum achievable frequency and r is normalized to $0 \leq r \leq 1$.

Eq. 2.4 can be used to estimate how power dissipation varies with processing rate. For example, if a frame of video requires half the maximum computation, the clock speed could be halved and voltage lowered to cut power by a factor of four: $E(.5) = .22 \times E(1)$.

Conventional digital logic works at a fixed voltage and idles if the computation finishes early. The comparable equation to Eq. 2.4 for fixed voltage is $E_{fixed}(r) = E(1)r$, because the energy per operation is constant and only the number of operations changes; the clock can be gated to avoid dissipating power once data processing has finished. So, to process the same half-frame of video, a fixed-voltage system uses 1/2 the energy it would take for a full frame, more than twice as much as a variable supply system!

Fig. 2-2 shows a plot of Eq. 2.4, for $V_{ref} = V_{DDmax} = 2V$ and $V_t = .4V$ along with the fixed-supply line. The ratio of the two curves — $\dfrac{E(r)}{E(1)r}$ — gives the energy savings ratio per block for any given r, but the ratio changes with r. At high rates the power is the same because the voltage is the same; at very low rates the variable voltage approaches V_t, so the ratio approaches is $\left(\dfrac{V_t}{V_{DD}}\right)^2$. The area between the curves is a convenient measure of the energy savings achievable by varying supply

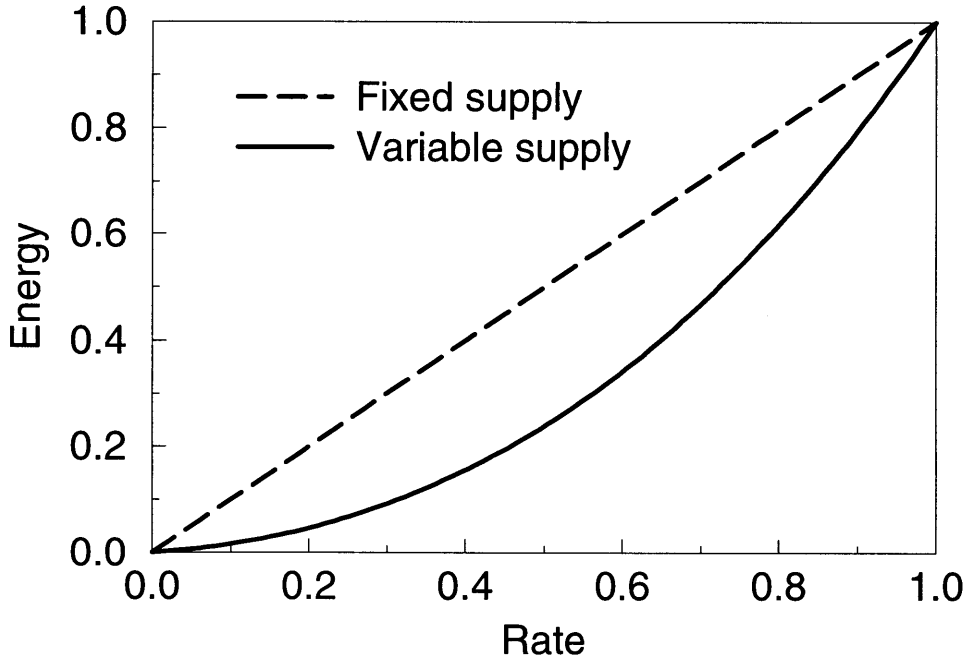

Figure 2-2: Energy *vs.* rate for variable and fixed supply systems

voltage without assuming a specific rate.

2.2 Second order corrections

2.2.1 Velocity saturation

One of the implicit assumptions made in deriving Eq. 2.2 was that the drift velocity of electrons in the channel is small compared to their thermal velocity. This corresponds to a V_{DS} much less than 1V per micron of channel length; for comparison, a digital circuit may have a 2V power supply and .3μm channel lengths, so a different model is needed to predict device behavior [13]. A precise calculation of velocity saturation influence on gate delay is messy [14], but a qualitative modification to Eq. 2.2 will suffice:

$$f = \frac{\alpha(V_{DD} - V_t)^2}{V_{DD}(1 + V_{DD}/V_S)} \tag{2.5}$$

V_S is the "corner voltage" at which velocity saturation effects limit the output current to half the value it would have been without the short channel effect. The extra

factor has the desired effect of lowering the the speed at supply voltages higher than V_S without affecting performance at low voltages.

This modified gate-delay equation yields a modified E-r relationship; the analog of Eq. 2.4 is

$$E(r) = E_0 r \frac{\left[\frac{V_t}{V_0} + r/2 + \sqrt{r\frac{V_t}{V_0}(1+\frac{V_t}{V_S}) + (r/2)^2}\right]^2}{1 - r\frac{V_0}{V_S}} \tag{2.6}$$

With exactly the same definitions as before; note that f_{ref} no longer corresponds to the frequency of operation at V_{ref}.[2]

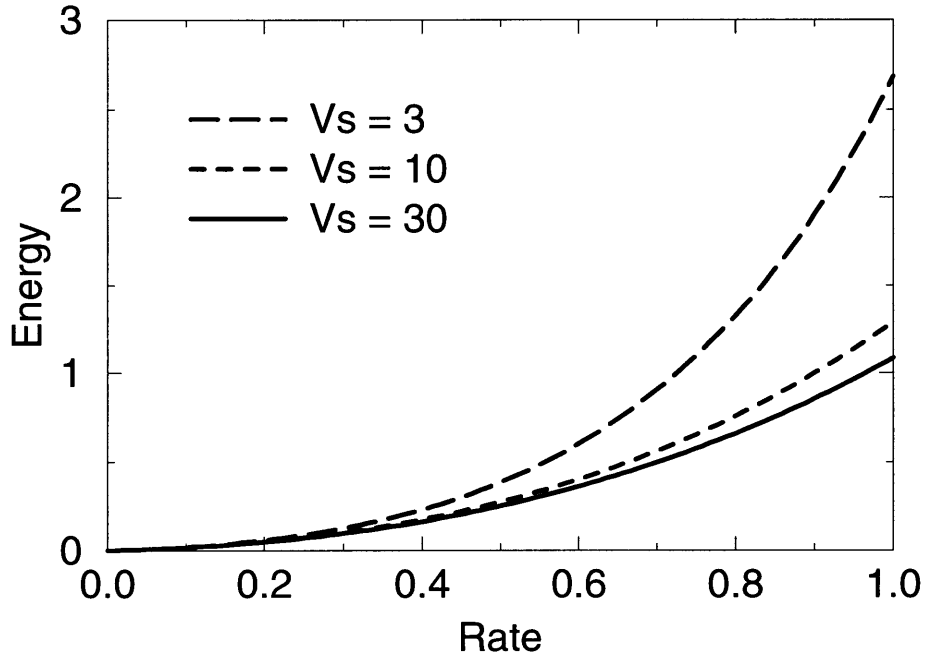

Figure 2-3: Effect of velocity saturation on the E - r curve

A curve showing the effects of velocity saturation is plotted in Fig. 2-3 for V_S = 3, 10, and $30V_{ref}$. At low voltages, the curves are identical, but as velocity saturation becomes noticeable, the energy for given rate rises rapidly. Qualitatively, this occurs because delay becomes less sensitive to supply voltage, so large voltage changes are needed to attain the desired frequencies. In terms of the framework developed in this chapter, the high curvature of the E-r function indicates that dramatic energy

[2] V_0 could be redefined to preserve the meaning of f_{ref}, then the normalization of r would depend on V_S; the current convention enables meaningful comparison of $E(r)$ *vs.* V_S.

savings are possible by scaling voltage, either statically by changing the architecture, or dynamically.

2.2.2 Leakage

Equation 2.1 does not describe all the power dissipation in ICs. One correction term, for example, is the power loss due to short-circuit currents during switching. For the sake of this discussion that term can be lumped into 2.1 by increasing C slightly, and the analysis remains essentially the same. On the other hand, power dissipation due to sub-threshold conduction is fundamentally different.

This leakage current adds a term to the DC power consumption that depends linearly on V_{DD} but *exponentially* on V_t:

$$P_{subthreshold} \propto V_{DD} \cdot e^{-V_t/nV_T} \tag{2.7}$$

where $V_T = kT/q$ is the thermal voltage and n is a non-ideality factor approximately equal to 2 for MOSFETS. As supply voltages scale down, threshold voltages scale down as well and leakage becomes a significant fraction of the total power [7]. Following the logic of the first section, it may well save power by slowing down — but slow down by raising V_t instead of lowering V_{DD}. Fig 2-4 shows the similarity between the methods.

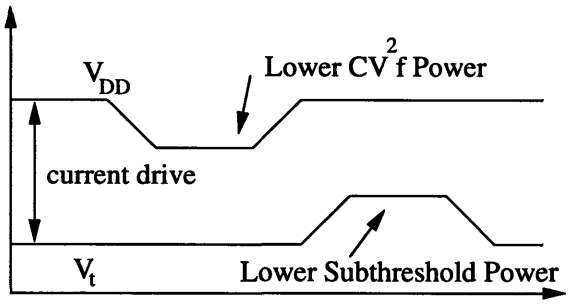

Figure 2-4: Raising V_t *vs.* lowering V_{DD}

The numbers bear this out: take a 1.2V supply voltage with .3V V_t, and assume that the activity is such that leakage contributes 30% to total power. Raising V_t by 50mV slows the circuit down by about 10%, and cuts the total power by .30 × (1 −

$e^{-50/(2 \cdot 26)}) + .1 \times .7 \approx 26\%$ which counts the 7% savings for doing fewer operations and the rest for the reduction in leakage. For comparison, varying V_{DD} for the same speed reduction gives only about 15% lower power.

Leakage power will not be addressed specifically in the rest of the thesis. However, almost all of the results derived for variably V_{DD} apply equally well to variable V_t. When this component of the power becomes significant, variable bulk biasing will be one way to lower power.

2.3 Parallelization

The plot of energy versus rate is a convenient tool to use to analyze the influence of different architectures on power. In particular, the effect of parallelization is easy to show. The idea is to copy a circuit block N times and divide the system clock by N; since the clock slows down, the voltage can be lowered, and the system runs at lower power. Algebraicly, the parallel system has an energy – rate relation given by

$$E_{par}(r; N) = (1 + k)NE(r/N) \tag{2.8}$$

where the $(1+k)$ accounts for overhead associated with parallelization. This overhead can be substantial, though the discussion is outside the scope of this thesis; $k = 0$ will be assumed. Fig. 2-5 shows an E-r plot of Eq. 2.8 for $N = 1$ (the reference system) and for $N = 2$ and 16, again with the fixed-supply lines for comparison.

There are two important to be made from the plot. First, increasing parallelization lowers energy for all rates (again assuming $k = 0$). The second is that the E-r curve has a smaller curvature for the higher Ns. Equivalently, the area between the constant supply and variable supply curves is relatively smaller, so *the power savings achievable by varying voltage diminish for higher Ns.* This is exactly the same effect that limits the energy savings from further parallelization: in both cases the energy savings arise because energy per clock cycle increases with speed. As V_{DD} approaches V_t circuit delay increases rapidly, so slowing the clock allows only small changes in

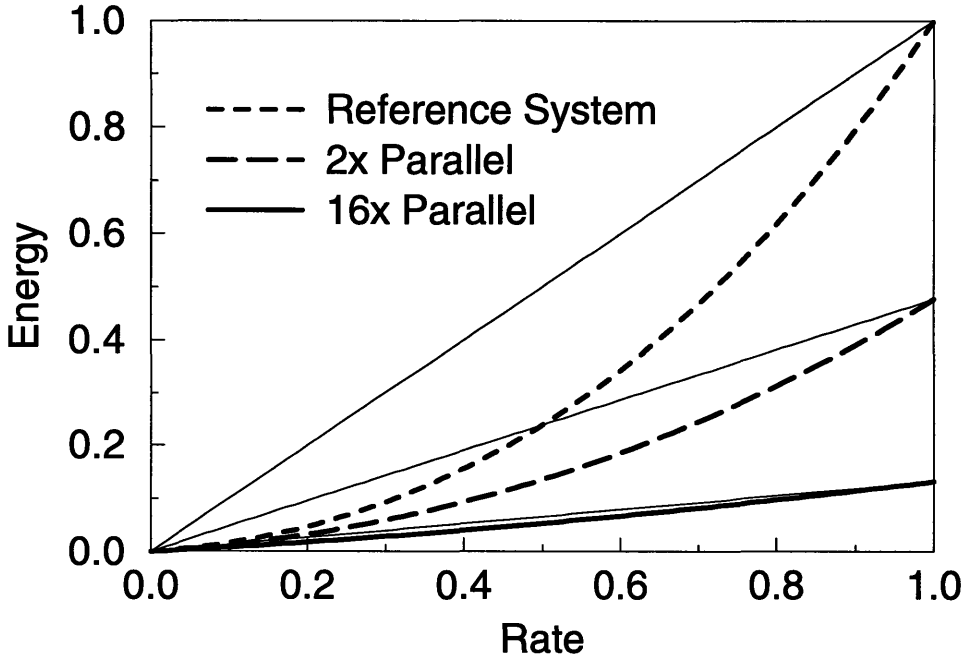

Figure 2-5: Effect of parallelization on power

V_{DD}, and if voltage doesn't change, the energy per clock cycle doesn't change with rate. Mathematically, this result comes naturally from Eq. 2.8:

$$\begin{aligned} E_{par}(r;N) &= NE(r/N) \\ &= E_0 r \left[\frac{V_t}{V_0} + r/2N + \sqrt{r/N\frac{V_t}{V_0} + (r/2N)^2}\right]^2 \end{aligned} \tag{2.9}$$

and taking the limit as N goes to infinity gives

$$\lim_{N\to\infty} E_{par}(r;N) = Cf_{ref}T_0 r V_t^2 \tag{2.10}$$

which is the expected minimum. In fact, because of subthreshold conduction Eq. 2.4 does not model device performance correctly if V_{DD} approaches V_t. However, gate delays increase so rapidly near the threshold voltage that Eq. 2.10 is nearly correct.

Chapter 3

Rate Control

The equations in the previous chapter define the static relationship between power and processing rate; in other words, the instantaneous power as a function of rate. The dynamic behavior, or the average power as a function of a sequence of rates, is equally important.

3.1 Averaging rate

There is a subtle but significant distinction between the processing rate r and what will be called the *computational workload,* denoted w. The rate r is the processing speed, or more simply the system clock frequency, while w is a measure of how much processing needs to be done on the incoming blocks of data. For example, a digital video application may have a maximum clock rate of 50MHz and computation dependent on what fraction of the image changed. Half of the image changing on a certain frame corresponds to $w = 1/2$. If the clock frequency happens to be 30MHz during the computation of that frame, $r = .6$; as long as both r and w are normalized, the computation on that sample will finish for $r \geq w$. Even in the cases where it is not necessary, it is often advantageous to buffer the workload so that r need not follow w exactly. Why incur the overhead of a buffer when r can be set equal to w?

3.1.1 Motivation

Consider three sequences of processing rates, $r_1[n]$, $r_2[n]$, and $r_3[n]$, with the following probability mass functions, or *pmf*s:

R	$Pr(r_1[n] = R)$	$Pr(r_2[n] = R)$	$Pr(r_3[n] = R)$
.1	.5	.25	0
.5	0	.5	1
.9	.5	.25	0

Note that the average rate is the same for all three distributions — $\overline{r_1} = \overline{r_2} = \overline{r_3} = .5$ — so in the long run the same number of operations is performed in all cases, yet the expected energies turn out quite different. Substitution into Eq. 2.4 shows that $\overline{E(r_1)} = .40$ while $\overline{E(r_2)} = .31$ and $\overline{E(r_3)} = .21$.

3.1.2 Convexity and Jensen's inequality

The dependence of power on pmf arises from the fact that energy is a *convex* (or "concave up") function of rate. Since both the energy per block and the number of operations done per block add linearly, the point describing the average rate and dissipated power for any two samples lie along the chord connecting their respective operating points; if the rate goes from .1 to .9, for example, the average power lies on the line connecting the points $[.1, E(.1)]$ and $[.9, E(.9)]$ as shown in Fig. 3-1. But since the Er curve is convex, any such chord always lies above the curve! So, if we average the workload and the power dissipated over two sample periods, the average power will be lower. In the example above, the sequence r_2 corresponds to r_1 averaged two blocks at a time, and indeed the power dissipation is lower.

Of course, this process can be continued further, simply by averaging workload over longer periods and lowering the energy further. In the limit, if we could average workload over all the samples first and then process them at a fixed rate, the average power would be at the minimum; in fact, $E(r_3)$ was much lower than $E(r_1)$. This

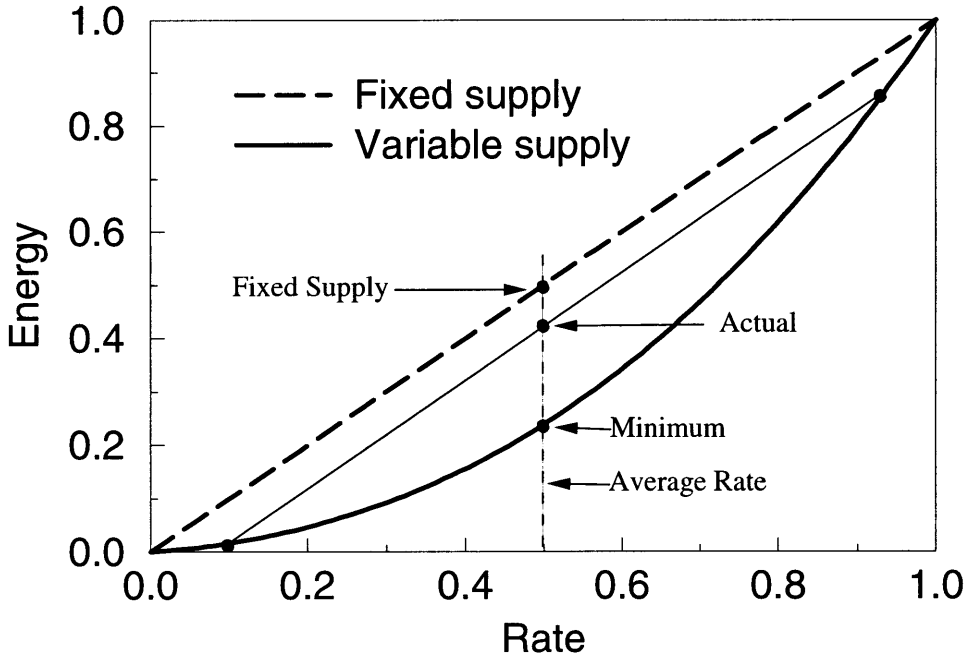

Figure 3-1: Averaging example.

result is summarized in *Jensen's inequality* [15].

$$\text{for convex } E(\cdot): \qquad \overline{E(r)} \geq E(\overline{r}) \tag{3.1}$$

In short, averaging the rate lowers power. It is this result that makes filtering the rate important for variable-supply systems; for comparison, note that the Er curve for fixed-supply systems is straight, so averaging the rate does not save power there.

3.2 Constraints

As argued above, the lowest possible power is achieved by averaging rate over all samples. In practice, latency and buffer size constraints limit the number of samples over which the computation can be averaged. Both constraints are invoked by assuming that there exists a finite buffer of size B that may not overflow as data arrives. Of course, the buffer cannot be allowed to underflow either — data cannot be processed before it arrives. Two more constraints need to be added to fully describe the problem: non-negativity and causality. The buffer over- and under-flow constraints on the

rate $r[n]$ given a workload $w[n]$ are

$$\forall n_0 : \qquad \sum_{n=-\infty}^{n_0} w[n] \geq \sum_{n=-\infty}^{n_0} r[n] \geq \sum_{n=-\infty}^{n_0-B} w[n] \tag{3.2}$$

The left inequality guarantees that no more data is processed than has arrived, *i.e.* that the buffer cannot underflow. The right one has the complementary role: to guarantee that all data that arrived B or more samples ago has been processed; that is, the buffer cannot overflow. Of course, in the long run, the average rate and the average processing load must be equal. This is consistent with the limits as stated; dividing the inequalities in Eq. 3.2 by $1/n_0$ and taking the limit[1] as n_0 goes to infinity gives $\overline{r[n]} = \overline{w[n]}$.

As described by the constraints in Eq. 3.2, the optimum $r[n]$ depends on the pmf of the workload. Even for simple distributions the optimization rapidly becomes unmanageable. Furthermore, even when a functional form exists, the calculation of $r[n]$ from a $w[n]$ sequence could be prohibitive in terms of area and power. Fortunately, approximations exist.

FIR filter of the workload

A moving average of the workload is a simple, good approximation to $r[n]$. It can be shown that any weighted average satisfies the required constraints for any $w[n]$. This average can be written as an FIR filter with constraints as shown below in Eq. 3.2.

$$r[n] = \sum_{k=0}^{B-1} a_k w[n-k] \tag{3.3}$$

$$0 \leq a_k \leq 1 \qquad \sum a_k = 1 \tag{3.4}$$

The filter is causal by construction. It is also conveniently time-invariant and linear. Non-negativity of $r[n]$ is guaranteed by choosing the a_k as prescribed in Eq. 3.4. That

[1]It is possible to come up with pathological distributions for $w[n]$ that have no mean. Eq. 3.2 could be rewritten in terms of finite moving averages to fix this problem. However, the problem is sufficiently obscure in this context that it will be ignored.

this satisfies the over- and under-flow constraints is shown below.

FIR mechanics

Before starting the proof, the difference between the workload in the FIR and the sample in the input buffer should be stressed. The data samples in the input buffer are processed at varying speeds depending on the current value of r. Sometimes more than one sample is finished per sample period, sometimes less; the number of data samples in the input buffer (*i.e.* the frames of video or digitized speech or whatever else) varies with time. On the other hand, the FIR filter performs a weighted average of the workload of the last B data samples, whether or not they have already gone through the DSP.

The difference is illustrated in Fig. 3-2. Entries in the "DATA BUFFER" represent data to be processed, with the denominator giving the total work needed to finish the sample and the numerator the work done so far. Note that the workloads here are not normalized to 1; the normalization is a matter of convenience. At $t = 3$, more than one sample is processed in one sample period. If only the workloads of the data samples in the data buffer were averaged, the rate would be lower than required by Eq. 3.3, and eventually the buffer would overflow.

FIR avoids buffer over- and under-flow

The proof proceeds by direct substitution of Eq. 3.3 into Eq. 3.2.

$$\begin{aligned}
\sum_{n=-\infty}^{n_0} r[n] &= \sum_{n=-\infty}^{n_0} \sum_{k=0}^{B-1} a_k w[n-k] \\
&= \sum_{k=0}^{B-1} a_k \sum_{n=-\infty}^{n_0} w[n-k] \\
&= \sum_{k=0}^{B-1} a_k \sum_{p=-\infty}^{n_0-k} w[p] \qquad\qquad p = n-k \qquad\qquad (3.5) \\
&\leq \sum_{k=0}^{B-1} a_k \sum_{p=-\infty}^{n_0} w[p] \\
&= \sum_{p=-\infty}^{n_0} w[p] \qquad\qquad (3.6)
\end{aligned}$$

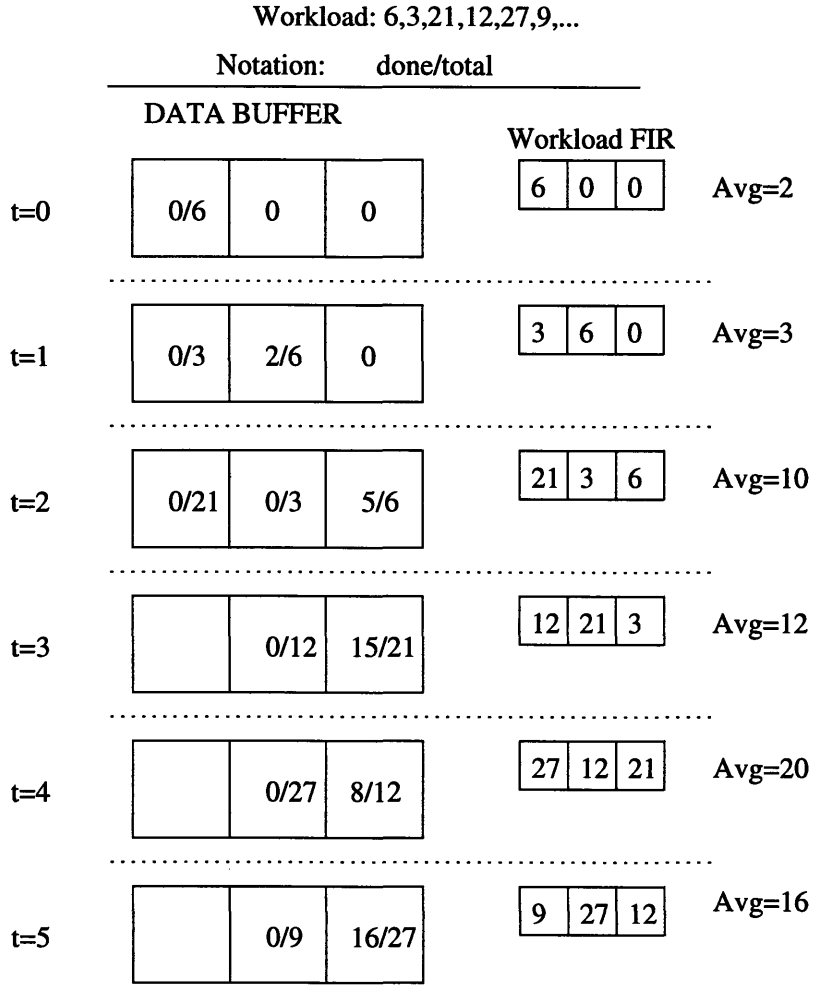

Figure 3-2: FIR mechanics

The inequality shows that the buffer cannot underflow, and a similar sequence shows that it cannot overflow. Starting from Eq. 3.5 gives the following result:

$$\begin{aligned}\sum_{n=-\infty}^{n_0} r[n] &= \sum_{k=0}^{B-1} a_k \sum_{p=-\infty}^{n_0-k} w[p] \qquad p = n-k \\ &\geq \sum_{k=0}^{B-1} a_k \sum_{p=-\infty}^{n_0-B} w[p] \\ &= \sum_{p=-\infty}^{n_0-B} w[p] \end{aligned} \tag{3.7}$$

Which completes the proof. Intuitively, the buffers do not overflow because the system is linear and any single data sample is processed correctly. That is, if only one data sample required computation and all the others were zero, by the Bth sample

time, exactly enough operations would have been completed to finish processing that sample. Since the $r[n]$ superpose, the system satisfies the processing constraints for all the data samples.

3.3 Update rate

It is clear that data buffers allow r to be somewhat decoupled from w; as shown above, the rate can be averaged over several samples to lower power. It is a small step to push the buffering idea further: if the power (and hence the rate) are averaged over B samples, shouldn't we only have to update r at $1/B$ of the sample frequency? As shown in chapter 4, slowing the update rate can save power in the power supply. Unfortunately, this can have unwelcome repercussions in the power dissipated in the DSP. Two examples are analyzed below: a subsampled FIR controller and quasi-Poisson queue.

3.3.1 Subsampled FIR

Fig. 3-3 compares the processing rates on a sample $w[n]$ sequence achieved by two $B = 2$ controllers: system A updates on every sample, while system B updates on every other sample. Label the workload of the data sitting in the ith position of the input buffer w_i. By setting the rate equal to the average workload in the buffer — $r = \frac{w_1+w_2}{2}$ — system A never idles, and at the same time guarantees that a there is always room in the buffer for the next data sample when it arrives. This is a specialization of the "weighted average FIR" controller analyzed above.

The added constraint on system B appears when $w_1 > w_2$, which happens in the example at $t = 4$. If the system works at the average rate, the first element will not be out of the queue before the next data sample arrives. In fact, to guarantee that the queue does not overflow the rate controller has to work at $r = max(w_1, \frac{w_1+w_2}{2})$. In order to work at the average rate, system B would need a buffer of length 3. This extra buffering is needed by any system that slows down the update rate; in order to update at $1/C$ of the sample arrival time, the buffer length is constrained to be

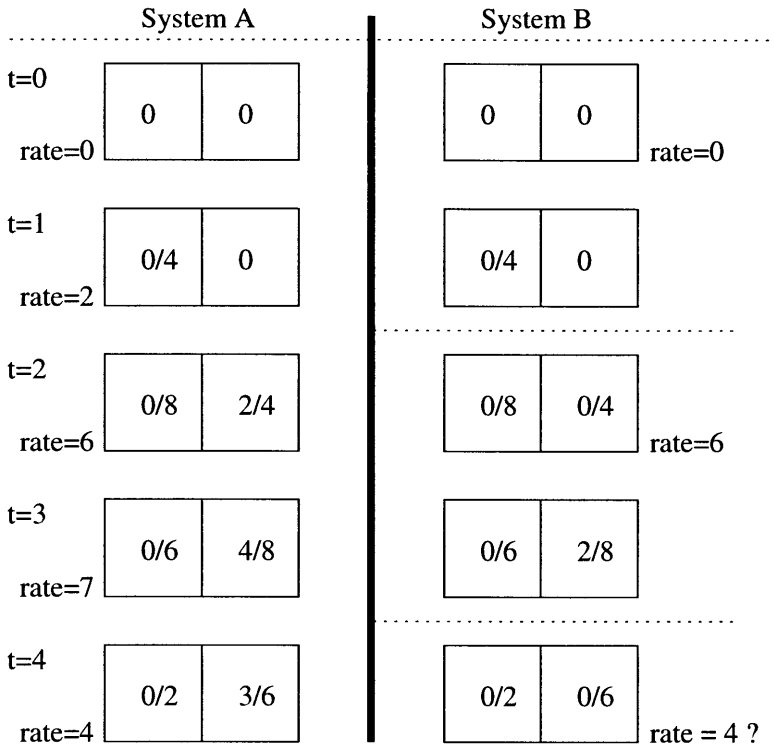

Figure 3-3: Rate averaging example

$B \geq 2C - 1$.

3.3.2 Quasi-Poisson queues

Poisson queues, or queues with exponentially distributed service and arrival times, may be used to model a variety of physical process, including some signal processing applications. For example, a packet-switched network may be modeled as a Poisson queue: the packets arrive independently, and are decoded and routed faster if they have fewer errors. We may be interested in finding the processing rate at which the probability P_L of losing packets due to buffer overflow is less that some constant. This problem is easily solved as follows.

Poisson queue solution

Define a state vector p where $p[i]$ gives the probability that i items are in the queue. The constitutive relationship for a Poisson queue is

$$\frac{dp}{dt} = Ap \tag{3.8}$$

where A is a matrix dependent on the service and arrival rates of the process. For a length 4 queue, A is given by

$$A = \begin{bmatrix} -\lambda & \mu & 0 & 0 \\ \lambda & -\lambda-\mu & \mu & 0 \\ 0 & \lambda & -\lambda-\mu & \mu \\ 0 & 0 & \lambda & -\mu \end{bmatrix} \tag{3.9}$$

where λ is the arrival rate and μ the service rate. In steady state, $\dot{p} = 0$. A bit of linear algebra gives

$$p \propto \begin{bmatrix} 1 & \frac{\lambda}{\mu} & (\frac{\lambda}{\mu})^2 & (\frac{\lambda}{\mu})^3 \end{bmatrix}^T \tag{3.10}$$

Any packet that arrives when the queue is full is lost, so P_L is given by

$$P_L = \frac{(\frac{\lambda}{\mu})^3}{1 + \frac{\lambda}{\mu} + (\frac{\lambda}{\mu})^2 + (\frac{\lambda}{\mu})^3} \tag{3.11}$$

The higher the processing rate, the lower the probability of losing a packet; for a buffer of length 4 and $P_L = .1\%$, $\frac{\mu}{\lambda} \geq 9.6$.

Variable rate processing

The high μ/λ ratio means that the packets must be serviced at a rate much higher than the average arrival rate, so the processor is idle nearly 90% of the time. This seems like an excellent application for a variable supply system: process quickly only when the queue starts to fill up.

Mathematically, such a system would behave as a queue with different service

rates in different states. Its A matrix would be

$$A = \begin{bmatrix} -\lambda & \mu_2 & 0 & 0 \\ \lambda & -\lambda-\mu_2 & \mu_3 & 0 \\ 0 & \lambda & -\lambda-\mu_3 & \mu_4 \\ 0 & 0 & \lambda & -\mu_4 \end{bmatrix} \tag{3.12}$$

with $\mu_4 \geq \mu_3 \geq \mu_2$. Again solving $Ap = 0$ we find that the steady state probabilities are

$$p \propto \begin{bmatrix} \mu_2\mu_3\mu_4 & \lambda\mu_3\mu_4 & \lambda^2\mu_4 & \lambda^3 \end{bmatrix}^T \tag{3.13}$$

The expected power[2] is given by $[0, {\mu_2}^2, {\mu_3}^2, {\mu_4}^2] \cdot p$. Optimizing the μ_i for minimum power (with the same .1% overflow constraint) gives

$$\mu_2 \approx 3 \quad \mu_3 \approx 7.5 \quad \mu_4 \approx 32 \quad \text{and } \overline{E} \approx 5 \tag{3.14}$$

Sampled queue

The system presented in Eq. 3.12 requires the rate to change at arbitrary times — any time a sample arrives or is serviced, the rate changes. A better model for a realizable queue would have the rate changing at discrete times.

Evolution of the state of the queue from one sample to the next is obtained by solving Eq. 3.8 for p. Just as in the scalar case, the solution is $p = p_0 e^A t$. To allow different processing rates based on the queue length at the sample epoch, each column would have its own μ, just as in the continuous-time case. It turns out that for this problem the matrix e^A can be computed symbolically for any size queue, but the result is extremely unwieldy; the matrix for a length-2 queue is shown below.

$$A = \begin{bmatrix} \frac{\mu_1+\lambda e^{-(\lambda+\mu_1)t}}{\lambda+\mu_1} & -\frac{\mu_2 e^{-(\lambda+\mu_2)t}-1}{\lambda+\mu_2} \\ -\frac{\lambda e^{-(\lambda+\mu_1)t}-1}{\lambda+\mu_1} & \frac{\lambda+\mu_2 e^{-(\lambda+\mu_2)t}}{\lambda+\mu_2} \end{bmatrix} \tag{3.15}$$

[2]The energy terms should be $E[\mu_i]$, where $E(\cdot)$ is defined by Eq. 2.4 rather than by ${\mu_i}^2$, but the optimization is much simpler with the simpler expression, and the results are essentially the same.

It becomes extremely difficult to optimize the rates symbolically for even a very small queue, so an approximate solution was obtained numerically. The minimum power solution for the length 4, .1% overflow case is plotted in Fig. 3-4 for a range of sample times.

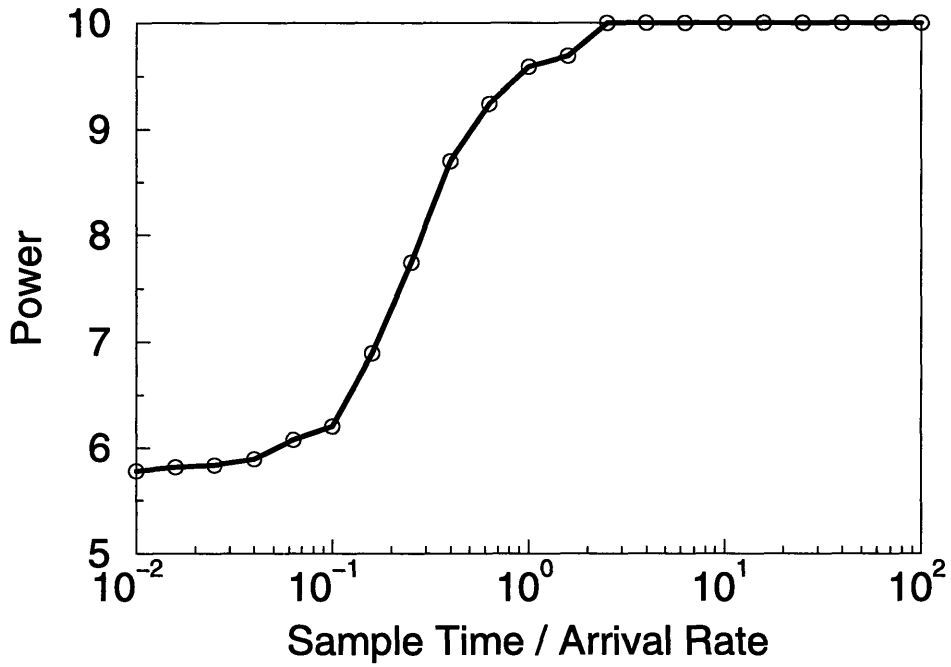

Figure 3-4: Minimum power *vs.* sample time

For update times much faster than the arrival rate, the minimum power solution approaches the unconstrained minimum of Eq. 3.14. For times much slower than the arrival rate, the system cannot vary the rate as fast as the queue fills up, so the minimum is simply the fixed-rate minimum. The key observation is that for supply voltage variation to be effective, the voltage update time must be comparable to the workload variation time. In other words, the bandwidth of the power supply must exceed the rate at which data arrives.

Chapter 4

Variable Supply and Clock Generation

In variable supply systems, power savings are achieved by lowering the supply voltage as the system clock slows down; indeed, this is the *only* reason such a system saves power. In the context of this thesis, the term "power supply" is used to mean the power converter that draws current from some battery or rectified source, filters and regulates the voltage level and outputs a supply voltage. The words "static" and "dynamic" will be used to describe the voltage level that is generated, *not* the internal operation of the converter; thus a switching converter that produces 5V will be considered static, and a linear regulator that can be set to 2V or 4V would be called dynamic.

4.1 Specifications

In the simplest static supply systems, the power supply is trivial — the battery voltage is used to power the chip directly. However, in most low power systems some regulation is required, and especially in the case of switching converters, low pass filtering is needed as well. The output voltage of dynamic converters also needs to be filtered and regulated, but the performance criteria are somewhat altered from those for a conventional, static power supply.

The first difference relates to the DC voltage level. A system with a static supply is typically designed to meet timing constraints at a specific voltage. At a lower level of abstraction, this means that feedback is established around the power converter to fix the output voltage as shown in the left half of Fig. 4-1.

A more efficient approach for fixed-rate systems was presented in [10, 9], where the feedback around the entire systems establishes a fixed circuit delay rather than a fixed voltage, as shown in the right half of Fig. 4-1.

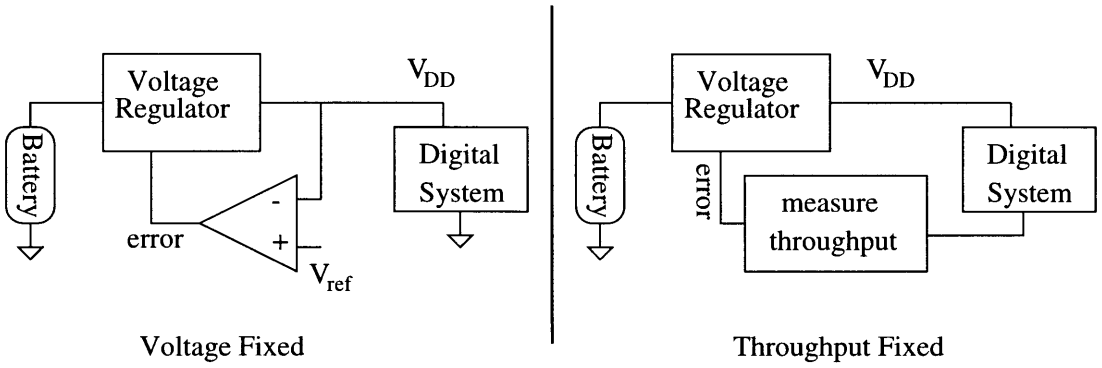

Figure 4-1: Feedback loops around the power supply *vs.* the system

The main advantage of the fixed-throughput approach is that the safety margin delay, the extra time allocated to make sure the chip meets timing constraints, needs only to account for small intra-die process variations since the inter-die delay variations are measured and compensated.

Fixed-throughput systems face a tradeoff between the efficiency gained by measuring throughput vs. the overhead of a variable power supply. Variable-rate systems already have the framework to change the supply voltage, so there is little reason to close the feedback loop around the power supply without including the circuit delays. Therefore, the DC level of the power supply becomes almost irrelevant.

Second, the desired transient response is markedly different. Since the ideal static supply has no variations on the output, the low-pass filter cutoff frequency is designed to be as low as volume and cost constraints allow. A dynamic supply still needs a low-pass filter to attenuate ripple, but also needs fast step response to allow rate changes as described in Chapter 3.

4.2 Supply topology

4.2.1 Linear regulator

The simplest and fastest power regulator available is the linear regulator. It consists of a variable-resistance pass device and a feedback loop that adjusts the resistance until the output voltage approaches a reference value, as shown Fig. 4-2. Both the advantages and disadvantages stem from the fact that this regulator is essentially a linear amplifier in feedback.

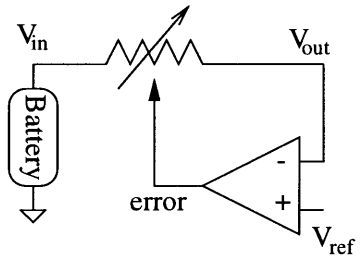

Figure 4-2: Linear regulator

First, the bandwidth of the regulator (from the reference voltage to the output) can approach several megahertz. This translates into microsecond or faster voltage changes, which can be useful if the sample frequency is high. An even bigger advantage is that the regulator can be integrated – no capacitors or inductors are needed to filter the output. And since it is linear, very little digital noise is generated during operation.

Unfortunately, these advantages are largely outweighed by the fact that the regulator is inherently lossy: current is drawn from the battery at a fixed voltage and delivered at a varying voltage to the load, with the voltage difference dropped across the regulator. For example if the output voltage is half the input voltage, half of the power drawn from the battery is dissipated in the regulator. The losses do not preclude the use of linear regulators in low power systems but do make the linear approach less attractive.

4.2.2 Switched capacitor converter

Switched capacitor charge pumps can also be used as power converters [5]. Power is transferred in a step-up (step-down) converter by charging two capacitors in parallel (series) and discharging them in series (parallel). The schematic (Fig. 4-3) is the same, and power flow is determined by placement of the source and load. Other configurations are possible to give any rational conversion ratio, though the ratio is fixed by the topology.

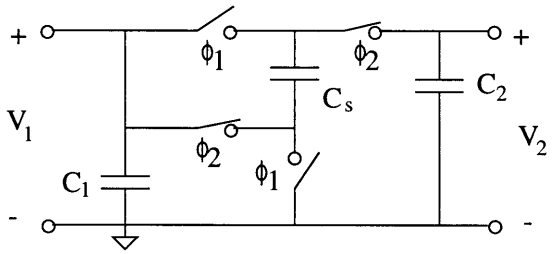

Figure 4-3: Switched-capacitor converter

For step-up operation, a voltage source (*i.e.* a battery) is placed on the low-V side and the load, typically modeled as a resistor or current source, on the high-V side. The capacitor C_s is charged to V_1 during ϕ_1, and then switched in series with C_1 during ϕ_2 to give double the voltage. A charge δQ flows into C_2, voltage across C_s drops by $\delta V = \delta Q/C_s$ and then the capacitor is switched back and the cycle repeats. Step-down operation is the same, except charge flows in the other direction.

Energy is lost in this converter when capacitors with different voltages are connected together: on ϕ_1 Cs is charged from $V_1 - \delta V$ to V_1 and during ϕ_2 it discharges back to $V_1 - \delta V$. This can be modeled as an effective resistance in series with the output of value equal to $R_{eff} = 1/f_s C_s$ where f_s is the switching frequency.

In principle, all of the components in the converter could be integrated. However, regulation is tricky (a different set of capacitors would be needed for each fraction of the input voltage), and to keep R_{eff} low off-chip capacitors are usually required.

4.2.3 Switching converter

Unlike a linear regulator, the switching converter is a sampled rather than a continuous time system. A representative switching converter called a *buck converter* is shown in Fig. 4.2.3.

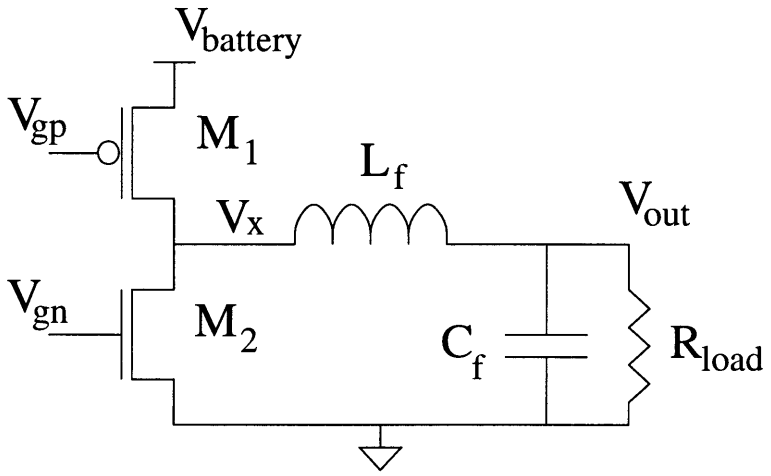

Figure 4-4: Simplified buck converter schematic

When transistor M_1 is on and M_2 off, voltage V_x rises to V_{DD}. Conversely, with M_2 on and M_1 off, V_x approaches ground. The duty cycle of the resultant square wave on V_x determines the DC level of the output voltage and the ripple is attenuated by an LC filter.

Unlike the linear regulator where losses were inherent in the operation, the power loss in the buck converter is due to parasitics — the losses approach 0 with ideal elements, and efficiencies of over 90% have been reported [16]. The major losses are due to series resistance in the inductor and switches, and the CV^2f power due to the gate and junction capacitance of the switches.

Both the series resistance and the gate capacitance of the switches depends on the channel length of the switching FET in the same way, so the L of the devices should be minimized. However, the resistance goes down with width and the capacitance increases, so there is a trade-off for W. A convenient result from the optimization is that at the minimum power point, equal power is dissipated in the series resistance of the switches and in driving the gate.

Active Damping

To prevent ringing on the output of the switching converter, the LC filter should be damped. In most cases, the series resistance of the switches or the parallel resistance of the load presents enough damping. If that is not the case, extra damping needs to be added to limit oscillations.

A simple approach that works well for fixed supplies is to add a resistor R_p in parallel with the load; to avoid the DC dissipation, a large capacitor can be added in series with R_p, as shown in Fig. 4-5.

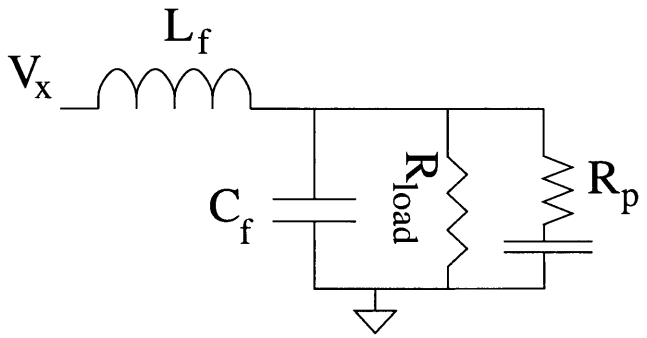

Figure 4-5: Passive damping

This turns out not be an efficient method for variable supplies because the parallel resistor dissipates energy every time the voltage changes. It is possible to avoid this by *actively damping* the loop; that is, introducing feedback into the system to emulate a resistor. The transfer function from the input to the output with a forward gain a and feedback gain f is $A = a/(1 + af)$. The forward path is a second-order filter, so a has the form $1/(s^2 + 2\alpha s + \omega_0^2)$ where α is a damping term and $\omega_0 = \sqrt{1/(LC)}$ is the resonance frequency. Hence, the total response is

$$\begin{aligned} a &= \frac{1/(s^2 + 2\alpha s + \omega_0^2)}{1 + f/(s^2 + 2\alpha s + \omega_0^2)} \\ &= \frac{1}{s^2 + 2\alpha s + \omega_0^2 + f} \end{aligned} \tag{4.1}$$

To increase the damping, the feedback needs to be of the form βs for some constant β; the current through a capacitor is of this form. A capacitor cannot be used directly because the feedback signal must interface to the digital part of the signal, so an A/D

converter is needed. Fig. 4-6 shows the block diagram of an actively damped supply. The current through the capacitor is digitized and passed back to the pulse-width modulator. The power cost of this scheme is in the A/D, but very low resolution is needed so the A/D can be small and low-power.

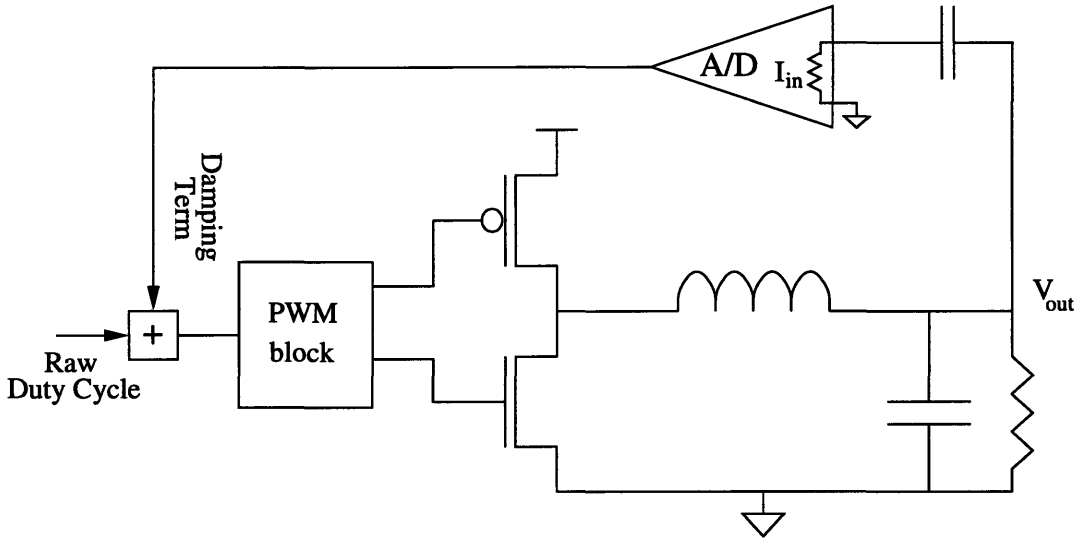

Figure 4-6: Active damping

Simulated waveforms are shown in Fig. 4-7. Without any damping the voltage would oscillate at each step; with active damping the circuit behaves just as it would if it were passively damped, except that the power dissipation can be lower.

4.3 Loop stability and step response

4.3.1 Open loop

Two methods to establish a voltage appropriate for a given rate have been mentioned in the previous section. The first is an extension of the static supply approach; for every desired rate an adequate voltage level would be read out of a ROM and passed to the voltage regulator, as shown in Fig. 4-8

In the case of a linear regulator the data would be passed to a D/A; in a switching regulator, to some sort of pulse modulator. When desired rate increases, the voltage goes up first to ensure that timing constraints will be met, and then the clock speeds

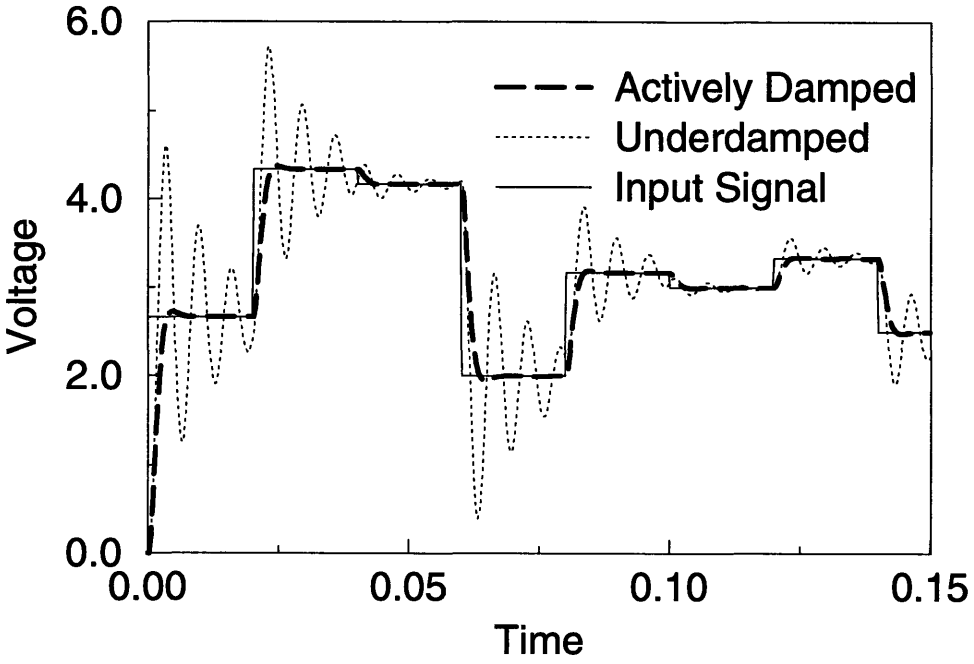

Figure 4-7: Active damping waveforms

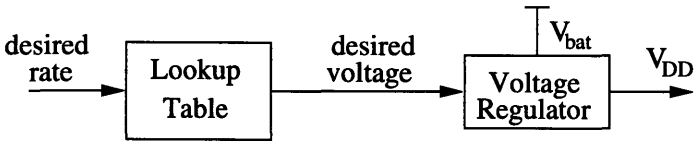

Figure 4-8: Open-loop variable supply system

up; when rate falls, the clock slows down and then the voltage drops. This has the advantage that changes in rate yield the fastest possible transitions on the output voltage, but as mentioned above, process and temperature variations would force the voltage to be high enough that delays meet worst-case, rather than actual, timing constraints.

4.3.2 Closed loop: PLL

The phase-locked loop approach avoids this problem. Rather than having a separate clock and power supply, both the clock and power converter are part of a feedback loop, and the system clock is based on chip delays. A system inspired directly by [10] is shown in Fig. 4-9.

Just as in the case of the fixed supply system, the PLL adjusts supply voltage to

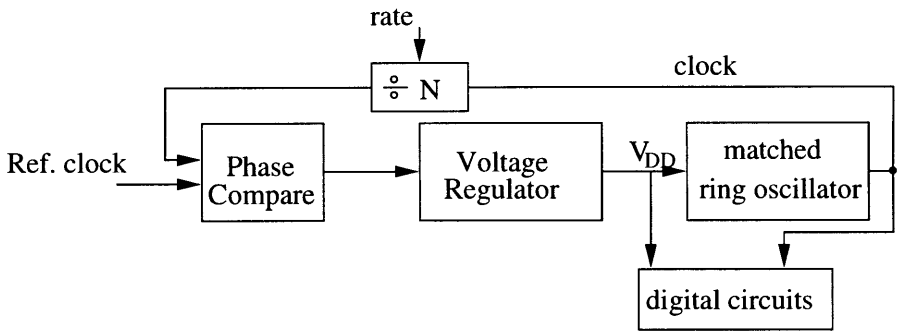

Figure 4-9: Block diagram of a phase-locked loop

the lowest possible level compatible with the required number of gate delays between registers. When the desired rate changes, the reference clock that the PLL sees changes and the loop re-locks. The drawback is in the time constant of the changes.

The time constant is determined by the bandwidth of the power supply, the loop filter, and the extra pole introduced by the integration of frequency to phase. With no loop filter, the feedback pole at 0 combines with the second-order pole[1] from the power supply to give a peaked response loop gain. A loop filter with a bandwidth significantly less than that of the power supply output filters can be added, but then this bandwidth limits how fast the loop locks.

4.3.3 Hybrid approach

The two previous schemes, the PLL and the lookup table, adjust the output voltage at different rates because they were originally intended for different functions. The PLL does best in tracking process variations where its low bandwidth is sufficient, and the lookup table is well suited to making fast voltage steps to predefined level. In fact, the characteristics are not mutually exclusive; it is possible to merge the lookup-table approach with the phase-locked loop to get fast voltage steps and process tracking at the same time.

The lookup table should still be used to get the fastest possible voltage changes; however, if the voltage levels are stored in a RAM instead of ROM they can updated to track process and temperature. A schematic is shown in Fig. 4-10.

[1] Assuming a switching supply with an LC output filter.

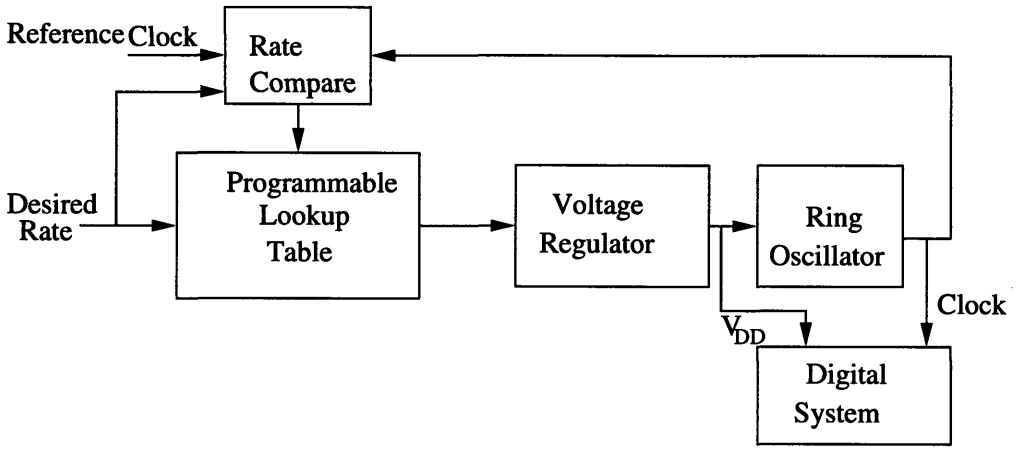

Figure 4-10: Hybrid system

The rate comparison and updates can be done very slowly compared to the bandwidth of the power supply. For example, if a buck converter switches at 1MHz, the output filters can have a bandwidth of $\approx$ 100kHz. For comparison, temperature compensation can be done at the frequencies below 1kHz. Thus, the dynamics of the power supply are insignificant in the feedback loop so there are no instabilities.

Quantization and dithering

Both the open-loop and the combined implementation have a lookup table to translate from rate to voltage. Since the overhead of the controller scales with the number entries in the lookup table, a smaller table is preferable to a larger one. Fig. 4-11 shows the Er curve for four-level voltage quantization. The lowest curve is the theoretical minimum E-r as predicted by Eq. 2.4; the area between it and the fixed-supply line is a measure of the power savings achievable by varying power.

If each sample must be processed at a fixed voltage and in one sample period (*i.e.* without workload averaging), the rate must be the next highest available rate. So, if the available rates are .25, .5, .75 and 1 and the sample workload is .6, the controller would have to choose the .75 rate and idle for part of the cycle. This gives the "stair-step" curve in Fig. 4-11.

If the voltage can be changed during the processing of one sample, the voltage can be dithered. In the example above, by processing for 40% a sample time at rate

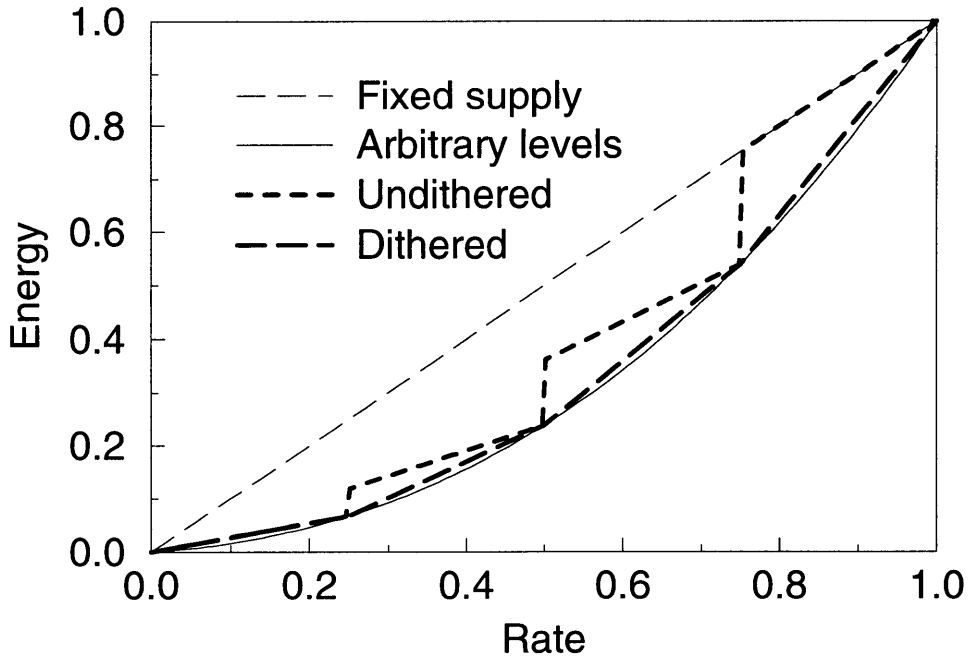

Figure 4-11: Quantization effect

of .75 and 60% at .5, the average rate can be adjusted to the .6 that is needed. This *dithering* leads to an Er curve that connects the quantized (E, r) points. As the figure shows, even a four-level lookup table is sufficient if dithering is used.

If the sample period is too short to allow dithering within the sample, the same effect can be achieved by allowing processing of one sample to extend beyond one sample period; in fact, this was the assumption made in the FIR filter analysis of section 3.2. If one sample is processed at a rate higher than its workload because of quantization, the next will be processed at a lower rate.

Chapter 5

Implementation and Testing

5.1 Chip design

A chip was designed to test the stability of the feedback loop and to verify that timing constraints are met as the modified phase-locked loop changes the clock and supply voltage. Since the focus is on the variable supply voltage control, only a token amount of processing is done by the DSP section, but it can be reconfigured to emulate applications with long sample periods (*i.e.,* computationally intensive cases like video processing) as well as applications with shorter sample periods. Similarly, the rest of the circuitry is designed for flexibility rather than efficiency.

5.1.1 Block diagram

The block diagram of the chip is shown in Fig. 5-1. At the highest level of abstraction, the test chip consists of four blocks. The FIFO and the DSP comprise the datapath, while the supply controller and ring oscillator are part of the control loop that generates both the supply voltage and the clock for the circuit.

During operation, input data is buffered in the FIFO until it is needed by the DSP. The control loop controls the processing rate to avoid queue overflow and underflow: as the queue fills up the clock speed increases to cope with the higher workload, and as the queue empties the clock slows. Thus, the FIFO acts as both a buffer for the

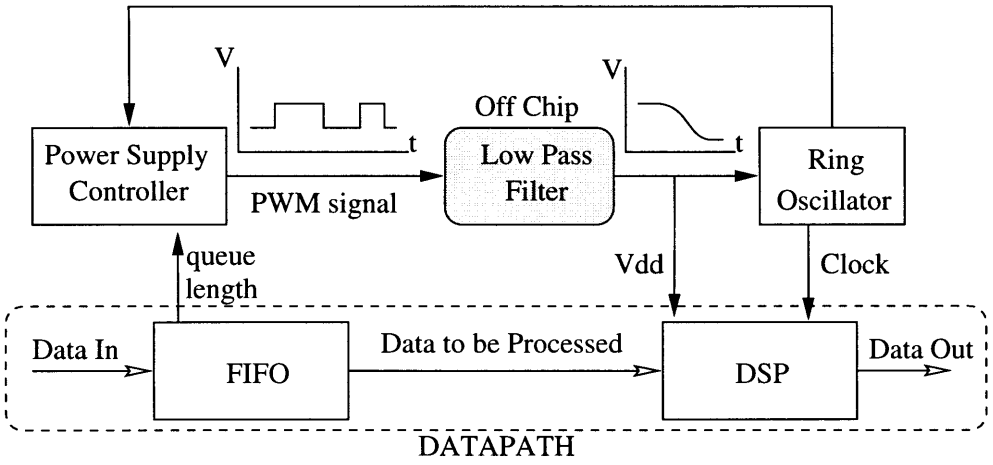

Figure 5-1: Dynamic supply voltage chip block diagram

data and as the workload-averaging mechanism. This control loop, which keeps the FIFO from overflowing by varying the processing rate forms the "outer" control loop.

A second, or "inner" control loop is formed between the power supply controller and the ring oscillator. This is the quasi-PLL that adjust the voltage steps in the controller to match processing rates. It is in this loop that temperature and processing variations are measured and corrected.

A schematic of the top level (and the other major blocks) is shown in the Appendix. The other structures at the upper level of hierarchy are involved with synchronization, buffering, and level conversions between the blocks.

5.1.2 FIFO Buffer

The FIFO is implemented as a 16 word by 4 bit data file with a counter to point to the head and one to the tail. The 4 bit data width and 16 word length was chosen to allow more precision in the queue length and workload than can be expressed in the level quantization; there is no other significance to the sizing of the buffer.

The block diagram is shown in Fig. 5-2. On the clock edge after the rising edge of **write**, data on the **datain** bus is written into the register and the head counter, **writeptr** is incremented. Since **write** comes from an input pin, it needs to be synchronized; the **synch** block generates a one-clock long pulse after a rising edge on its input.

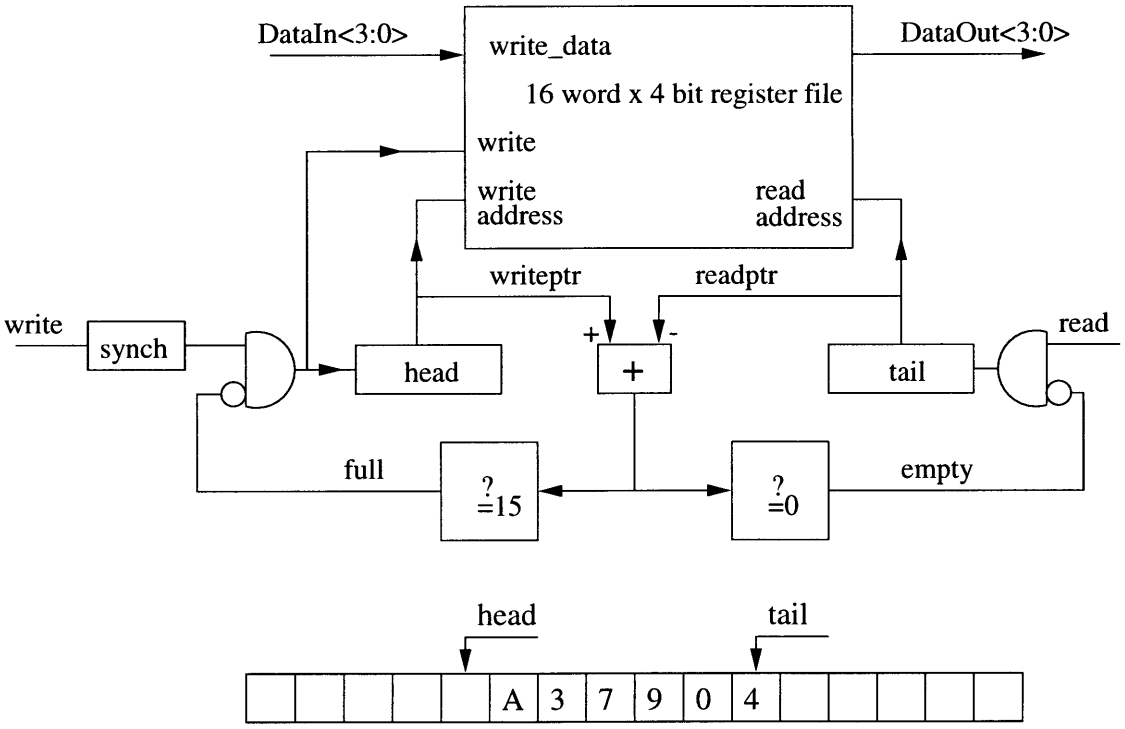

Figure 5-2: FIFO block diagram

The **read** signal is generated internally and is synchronous, so no other logic is necessary — if the queue is not empty, the tail counter, **readptr**, is incremented when **read** is asserted. Unless the queue is empty, data on the output is always valid — the read and write cycles use different busses and are completely independent.

The difference writeptr – readptr gives the number of words stored in the queue at any time; this signal is output as **qlen**. One gate generates the **full** flag which inhibits writing and disables the write counter and another the **empty** flag which disables the read counter. Data that arrives when the queue is full is lost.

5.1.3 Supply control

The supply controller has several functions: it assigns an operating rate based on workload; it translates that desired rate into a digital word proportional to the voltage level; and finally it generates a pulse-width modulated signal that is filtered and buffered off chip and fed back as the supply voltage for the DSP and ring oscillator.

Rate control

The desired rate is periodically updated based on the number of words in the queue. Specifically, the high two bits of **qlen** are latched on after the rising edge of **write**, which is generated externally and synchronized by the *synch* block. This is a direct implementation of the sampled queue controller described in section 3.3.2; workload is averaged implicitly by considering the queue length in the FIFO as the processing rate.

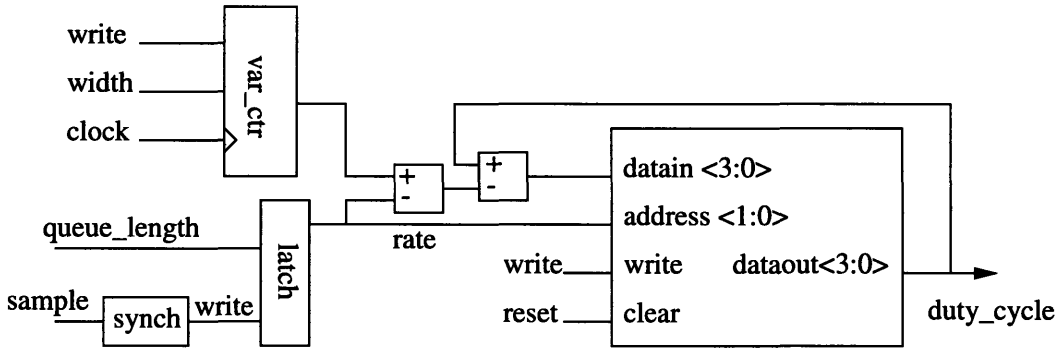

Figure 5-3: Rate translation block diagram

Voltage translation

The desired processing rate is translated to a supply voltage by means of a lookup table; a block diagram of this, together with the rate latch is shown in Fig. 5-3 The desired rate forms the address, and the 4 bit output gives the fraction of the battery voltage needed to achieve the desired rate. On **reset** the table is cleared to all zeros; on **write** the current entry is updated, so eventually the correct values are stored in all entries of the table. Since the table is updated periodically, the voltage levels track temperature and battery voltage (as well as process variations).

Updates are done by comparing the actual clock rate to the desired rate. The *rate counter* is reset on **write**, and counts "tics" for the duration of the sample. The *rate counter* is physically a 16 bit counter, but the **width** signal controls which group of four bits is output, so a tic can correspond to either 1, 16, 256, or 4096 clock cycles. The desired rate is translated to tics by multiplying by 2. Different settings allow for

faster or slower updates.

The count at the end of the sample period (*i.e.* at **write**) is subtracted from the desired rate to get the rate error. The error is then added the the current lookup table entry before the address changes. So, if the rate was too high (low), the error is negative (positive) and the voltage entry is lowered (raised). In the event of over or underflow, the value entered is limited to all ones or all zeros, respectively.

For example, the entry for rate = 3 in the lookup table might be 14. If the actual tic-count at the end of the sample period were 7, the error — $2 \times \text{rate} - 7 = -1$ — would be added to the lookup table entry, and the new value would be 13. If, on the other hand, the tic-count at **write** were 4, the error would be $6 - 4 = 2$ and the new entry should be 16; since that overflows the 4-bit words of the lookup table the actual entry is limited to 15.

PWM counter

The 4 bit word output from the register file is converted into a pulse-width by means of a counter. This conversion could be accomplished by running a 4 bit counter continuously, setting the pulse to 1 when the count crosses 0 and resetting the pulse to 0 when the count crosses **dcout**. This turns out to be a poor solution because for low **dcout**, the DC level of the waveform (and hence of the supply voltage to the ring oscillator and DSP) is so low that the clock stops. In fact, the controller could be designed to recover gracefully after the clock stops for a period of time, but any dynamic logic in the DSP would discharge after even a few microseconds without a clock. Therefore, it is necessary to keep the DC level above a minimum level at all times.

This is accomplished by using a 5 bit counter so that on every cycle the output pulse would be high for several clock periods regardless of **dcout**, with the pulse length beyond the minimum determined by **dcout** as shown in Fig. 5-4.

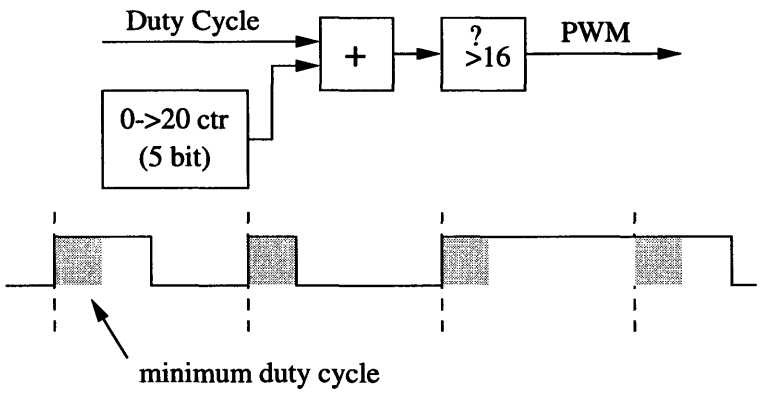

Figure 5-4: PWM block diagram

5.1.4 Ring oscillator

In a variable supply system the ring oscillator sets the clock speed. It is therefore crucial that it be well matched to circuit delays. In this case both the DSP "critical path" and most of the ring oscillator is a string of long inverters, so matching is not a problem. On the other hand, it is important to be able to change the period of the oscillator to verify the tracking.

Another concern relates to the power and area consumed by the ring oscillator. A ring oscillator made from a few long inverters is more efficient than one made from many minimum-size devices. In this case the minimum number of inverters was limited by the need to incrementally change the period — a three inverter oscillator may be ideal, but it allows little resolution in clock period.

The ring oscillator in the chip consists of nine inverters, with taps from the last four stages feeding into a mux and then an XOR (to keep an odd number of inversions around the loop) and back to the input, as shown in Fig. 5-5. This allows control of the clock delay in $\approx 10\%$ increments.

5.1.5 DSP

The only function of the DSP on the test chip is to emulate the throughput and timing constraints of a general variable-workload signal processor. This functionality was implemented as two separate parts.

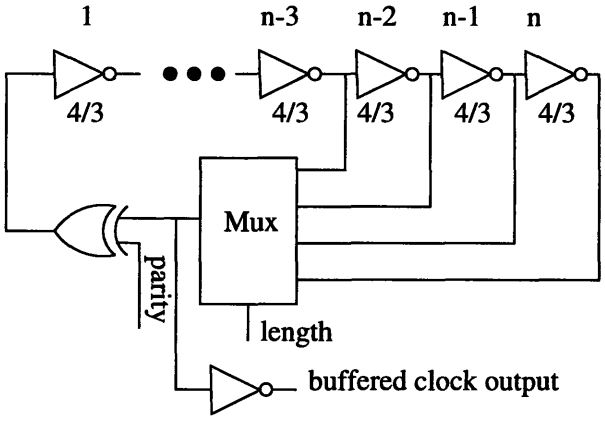

Figure 5-5: Ring oscillator block diagram

Any algorithm that terminates on a data-dependent condition generates a variable workload. For example, an algebraic approximation iteration that terminates when the desired precision is reached, or a video decompression procedure that processes data until it finds an end-of-frame marker both appear as variable workload algorithms. One of the simplest algorithms is a counter: it counts from zero until the count matches the input data. When the count matches, the data word is considered "processed," and another word is fetched from the FIFO. To verify that the right words are being fetched and executed, the counter state is an output of the chip.

Ideally, the variable supply chip could emulate both the computationally long samples common in video processing and the very short samples typical in communications systems. The four bit input words allow sixteen levels of workload for the DSP. The sixteen levels are sufficient to exercise the DSP at a fixed sample complexity, but are completely inadequate to span the orders of magnitude complexity needed. Rather than increasing the length of the data word (and with it, the buffer size) the complexity per sample is controlled by selecting one of four sets of four bits from the sixteen bit counter get treated as the "processing." The block diagram of the DSP is shown in Fig. 5-6.

The critical path consists of strings of long inverters placed between the counter and output latches. Since timing errors can only occur when the input is changing, the input to these delay chains comes from the lowest four bits of the sixteen bit DSP counter, independent of which bits actually form the "output" of the chip. Thus,

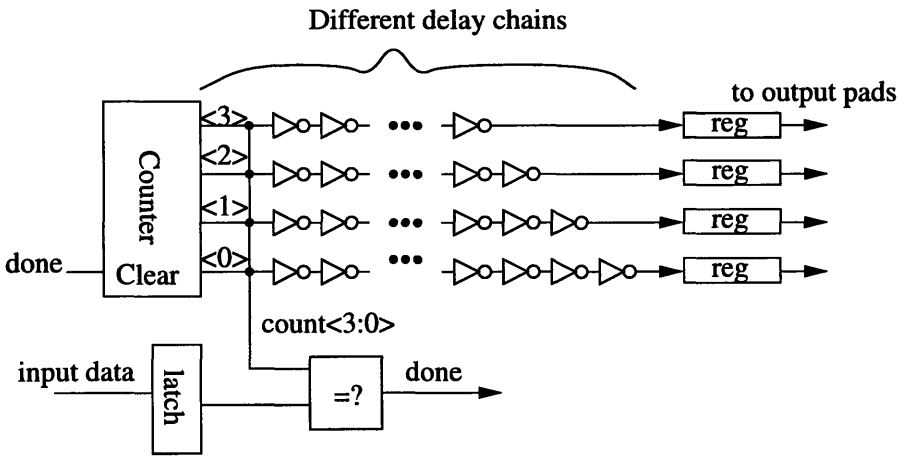

Figure 5-6: DSP block diagram

even if the output counter changes infrequently, the timing is checked on every clock cycle.

The four delay lines actually have different numbers of inverters. The difference allows calibration of the safety margin available in the clock cycle: if all delay lines shorter than a cutoff length consistently latch correctly and all longer lines do not, the clock is correctly tuned to model delays of the cutoff length. If errors are intermittent, the delay is not well matched.

5.2 Initial test results

The chip was fabricated in a .8μm CMOS process. The final layout consumed 2.2 $\times$ 3mm; the area was determined by the 40-pad pad ring. The chip photo is reproduced in Fig. 5-7.

Initial testing has verified basic functionality of both the DSP and the control loop. Fig. 5-8 shows measured waveforms for the two bits of the queue length and the supply voltage. The top two bits of the queue length are used to determine what rate to process at. As the figure shows, when the queue is full, the controller raises voltage to a level determined by the delays, and as the queue empties the voltage drops in discrete levels.

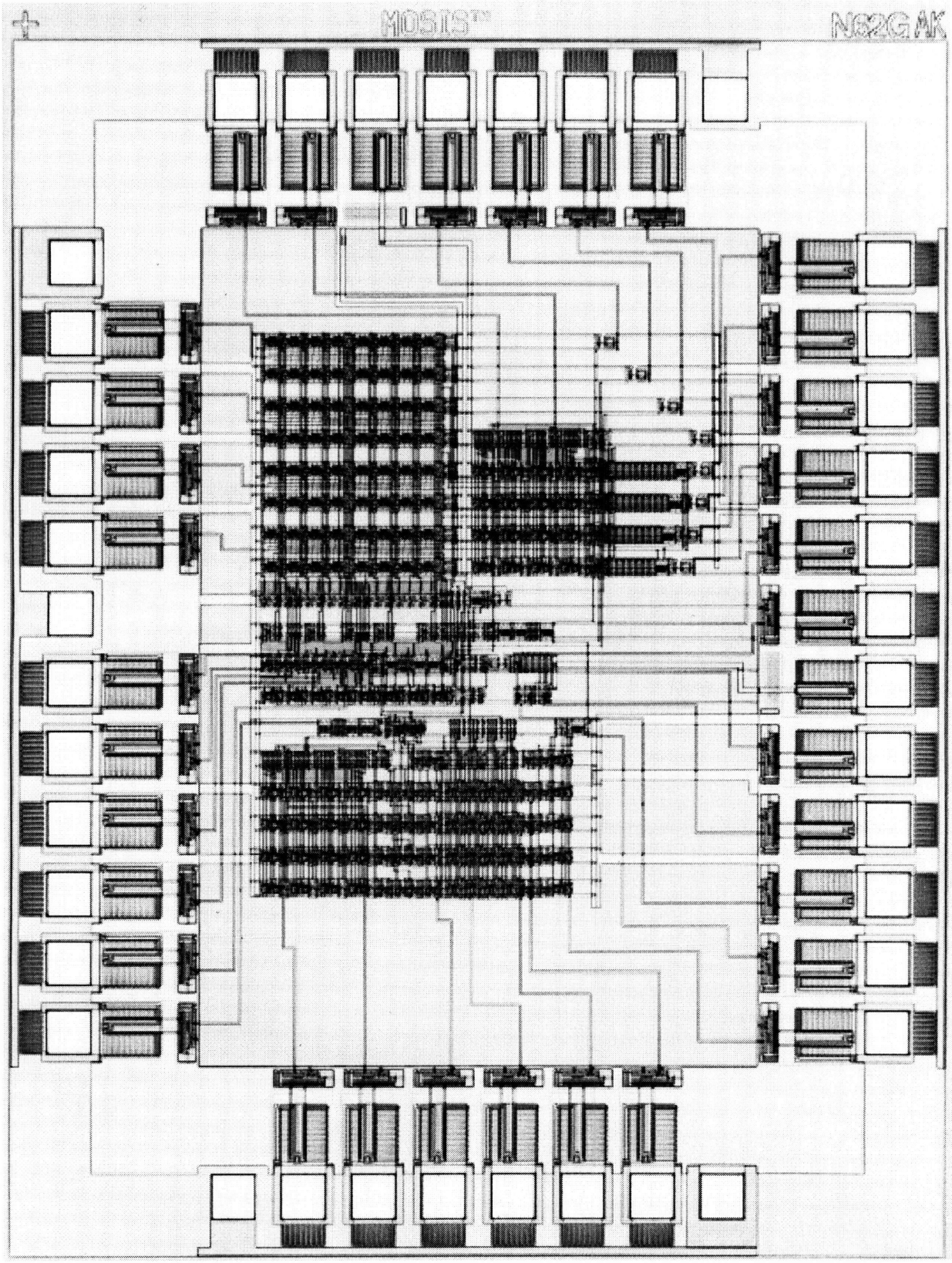

Figure 5-7: Variable supply controller chip plot

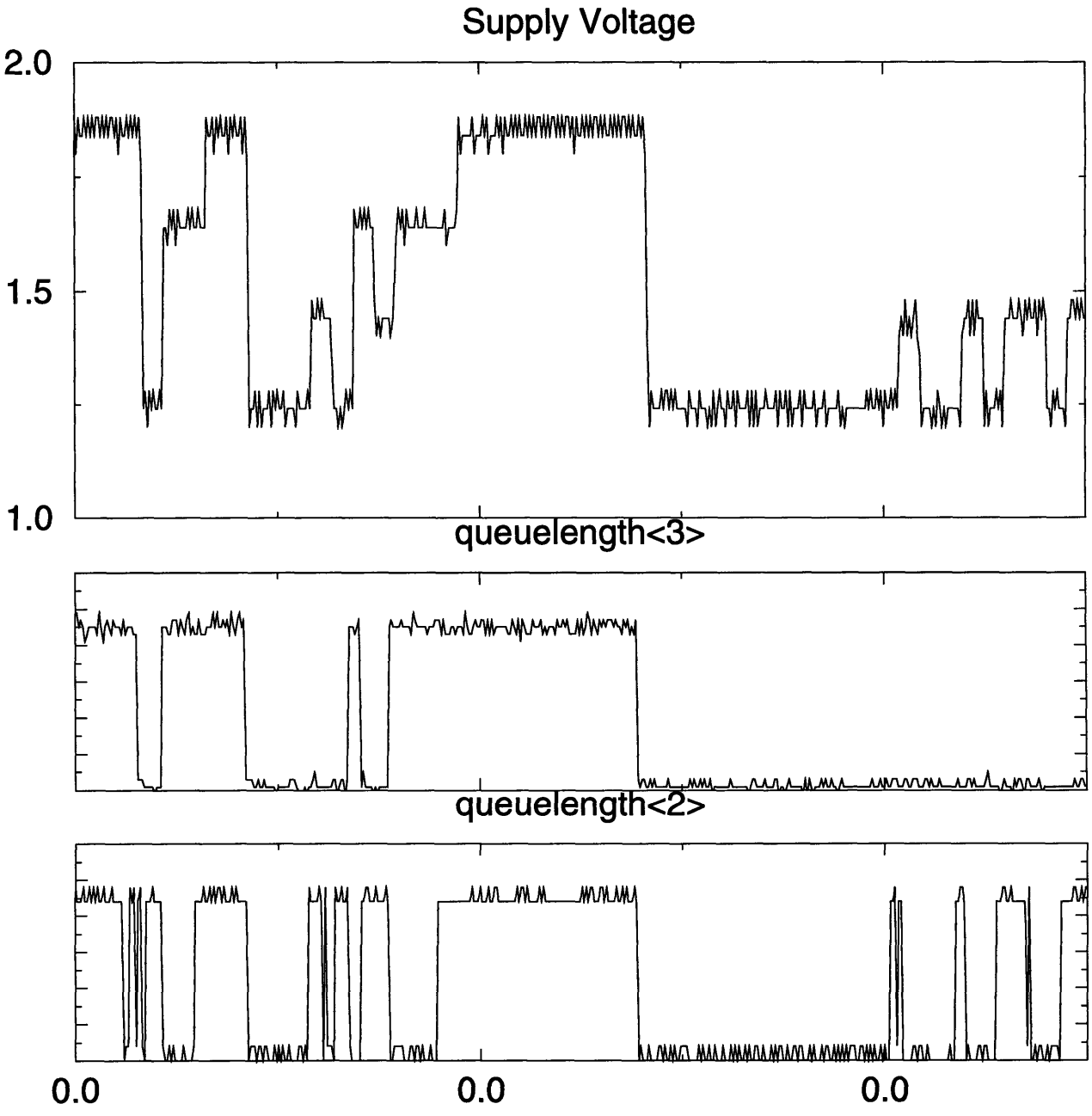

Figure 5-8: Initial results

Chapter 6

Conclusions

For applications where workload changes with time, the power consumed by the DSP can be lowered by varying supply voltage. Video and communications ICs are particularly likely to benefit from variable voltage supplies. The possibility of varying voltage dynamically may allow some fixed-workload algorithms to be replaced by variable-workload algorithms with lower mean computation requirements.

Where latency and extra buffering can be tolerated, averaging workload can lower power even further. When the workload can be predicted *a priori* explicit low-pass filtering can be done; otherwise, the data can be queued and processing adjusted on the basis of queue length. Optimum filtering algorithms are difficult to find and dependent on the statistics of the input data, but easily computed approximations exist.

Continuous variation of the supply voltage can be approximated by very crude quantization and dithering. As long as the power supply is damped correctly the voltage transitions do not cause extra dissipation. Since the power and area of the supply control circuitry scales with the number of voltage levels, quantization to only a few bits minimizes overhead. So few levels are required that it may even be possible to use an integrated switched capacitor regulator in very low power applications.

Finally, the control circuitry developed is applicable to general DSP circuitry. No assumptions have been made about the DSP in the development of the control; the DSP is synchronous. All that is required for the control is that the critical timing

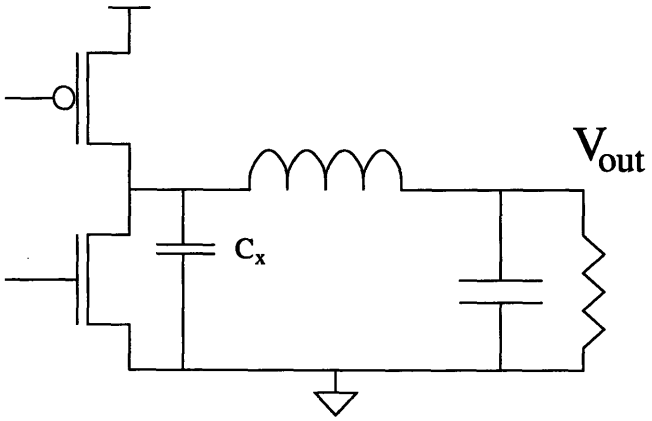

Figure 6-1: Parasitic losses in buck converter

path is known.

6.1 Variable supply tradeoffs

The only advantage of variable supply systems is the lower power. The tradeoffs required in the power supply to achieve this power savings are included in the calculation. However, there are several secondary drawbacks that a variable supply system necessarily suffers. This section discusses some of the drawbacks specific to variable supply systems.

A few circuit optimization techniques depend on a fixed supply voltage. For example, it is possible to equalize rise and fall times in static gates by sizing the devices properly; the exact sizes depend on the supply voltage. More significant effects are felt by the power converter.

Power supplies

One of the losses associated with switching power supplies is due to the CV^2f loss in the parasitic capacitance on the high-frequency output node; in Fig. 6-1 this is the capacitor C_x.

In the case of large switching devices, C_x is dominated by the drain capacitance of the switching devices. A technique to eliminate this loss called "soft switching" relies on adjusting the drive waveforms to avoid discharging C_x through the drive devices

[16]. By adjusting the delay between turning off one of the switches and turning on the other one, it is possible to turn on the FETs only when they have nearly $0V$ V_{DS}, thus eliminating switching losses.

Unfortunately, the exact timing depends on the current flowing in the inductor. If the output voltage is constant, the feedback loop suggested in [16] can adjust the timing. In a variable supply system the current changes with every voltage change, so there would be an unavoidable loss with every transition while the loop re-locks. If voltage changes frequently, soft switching becomes untenable.

6.2 Future research

This thesis describes the framework for implementing a variable power supply system. Specifics of the implementation would need to be investigated for each particular system; the approximations needed by a 10W system are completely different from those needed by a 10mW system. However, there are several issues specific to variable supply systems in general that need to be developed further.

6.2.1 Testability

Testing of a variable voltage system is somewhat more involved than that of a fixed supply system: not only does the functionality have to be verified for all the internal states of the system, but also for a range of voltages. So, a DSP chip that works at 3V may not work correctly at 2V if the clock generation was misdesigned or misfabricated.

6.2.2 Interaction of multiple systems

The thesis has assumed that a variable voltage IC would be embedded in a fixed-clock system. It may be the case that for large enough systems, it will be useful to put several different variable voltage chips together into a larger block. Another possibility is that several subsystems with separate variable supplies would be integrated onto a single chip. In either case, the interactions of the chips should be explored.

Specifically, the data rate at the output of a variable clock system is not the same as the input data rate. For proper operation, a macro-controller may need to be developed to manage power consumption of the whole system.

Bibliography

[1] Vincent R. von Kaenel *et. al.* . Automatic adjustment of threshold and supply voltages for minimum power consumption in CMOS digital circuits. In *IEEE Symposium on Low Power Electronics*, pages 78–79, 1994.

[2] Lars S. Nielsen *et. al.* . Low-power operation using self-timed circuits and adaptive scaling of the supply voltage. *IEEE Transactions on VLSI*, 2(4):391–397, December 1994.

[3] L. G. Heller and W.R. Griffin. Cascode voltage switch logic: A differential CMOS logic family. *ISSCC Digest of Technical Papers*, pages 16–17, February 1984.

[4] Gordon M. Jacobs and Robert W. Brodersen. A fully asynchronous digital signal processor using self-timed circuits. *Journal of Solid State-State Circuits*, pages 1526–1537, December 1990.

[5] Anantha Chandrakasan and Robert Brodersen. *Low Power Digital CMOS Design.* Kluwer Academic Publishers, 1995.

[6] R. M. Swanson and J.D. Meindl. Ion-implanted complementary MOS transistors in low-voltage circuits. *IEEE Journal of Solid-State Circuits*, 7(2):146–152, April 1972.

[7] M. Mutoh *et. al.* . 1-V power supply high-speed digital circuit technology with multithreshold voltage CMOS. *IEEE Journal of Solid State Circuits*, pages 847–854, August 1995.

[8] K. Seta *et. al.* 50% active-power saving without speed degradation using standby power reduction (SPR) circuit. In *IEEE International Solid-State Circuits Conference*, pages 318–319, 1995.

[9] Marc Horowitz. Low power processor design using self clocking. ARPA Low Power Electronics Annual Review, April 1995.

[10] Peter Macken *et. al.* . A voltage reduction technique for digital systems. In *Digest of Technical Papers, IEEE International Solid-State Circuits Conference*, pages 238–239, February 1990.

[11] J. Ludwig, H. Nawab, and A. Chandrakasan. Low power digital filtering using approximate processing. *IEEE Journal of Solid-State Circuits*, March 1996.

[12] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen. Low-power CMOS digital design. *IEEE J. Solid-State Circuits*, 27(4):473–483, April 1992.

[13] C. Sodini, P. K. Ko, and J. L. Moll. The effect of high fields on MOS devices and circuit performance. *IEEE Transactions on Electron Devices*, 31:1386–1393, Oct 1984.

[14] M. Kakumu and M. Kinagawa. Power-supply voltage impact on circuit performance for half and lower submicrometer CMOS LSI. *IEEE Transactions on Electron Devices*, 37(8):1902–1908, August 1990.

[15] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* Wiley Series in Telecommunications. John Wiley & Sons, 1991.

[16] A. Stratakos, S. Sanders, and R. Brodersen. A low-voltage CMOS DC-DC converter for a portable low-powered battery-operated system. *PESC*, 1994.

Appendix A

Schematics

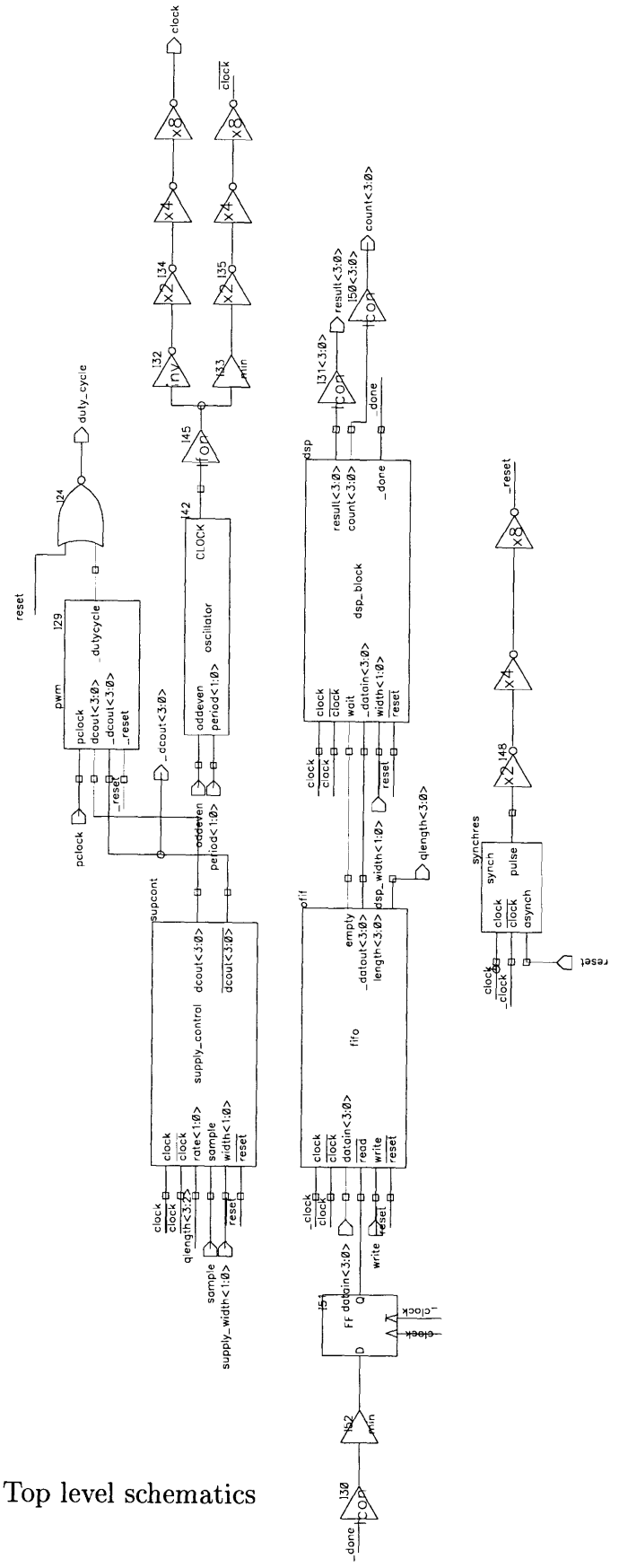

Figure A-1: Top level schematics

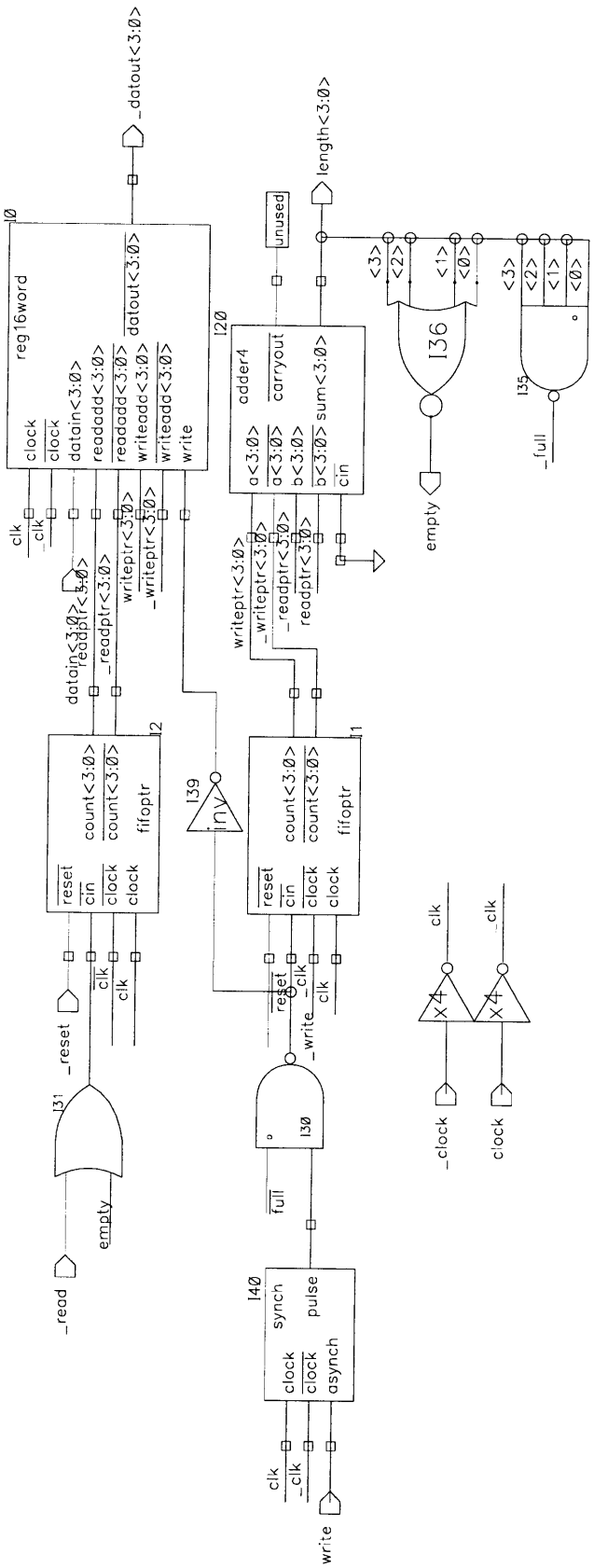

Figure A-2: FIFO buffer

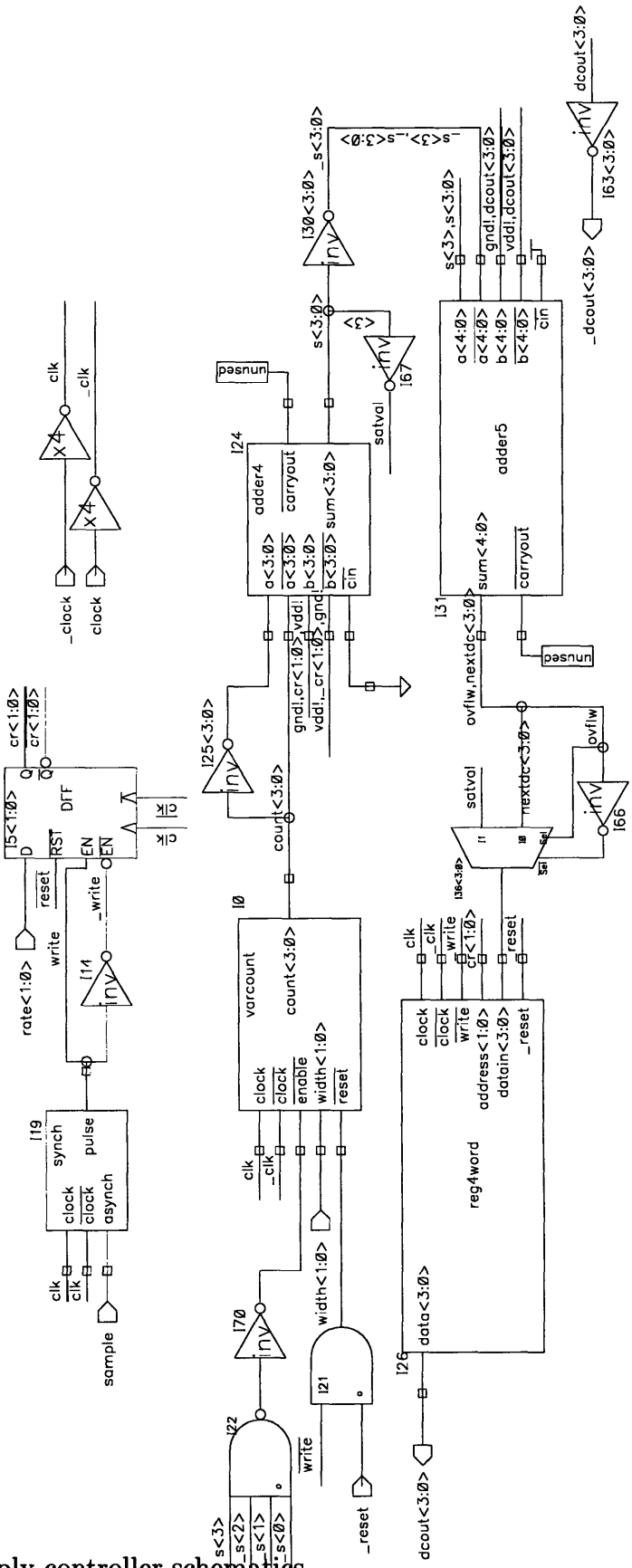

Figure A-3: Power supply controller schematics

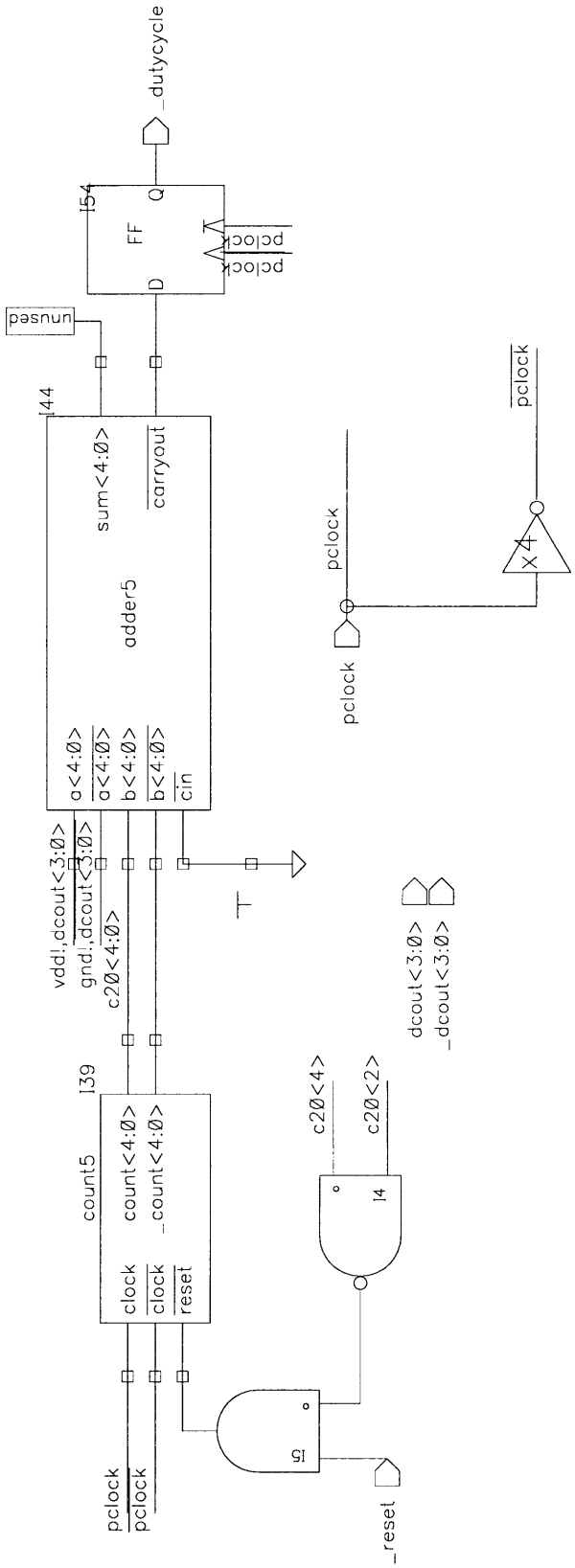

Figure A-4: Pulse width modulator

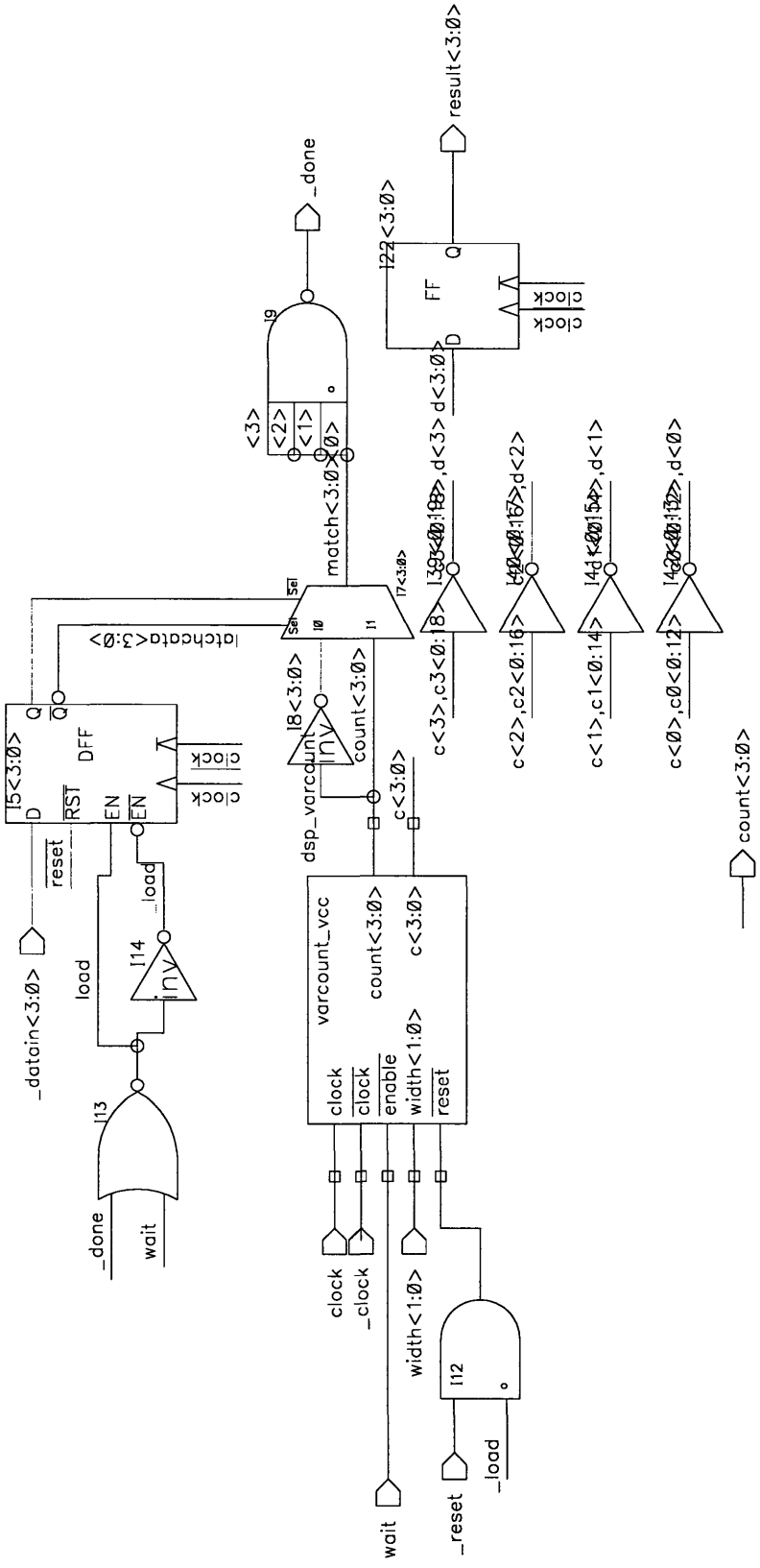

Figure A-5: Signal processing and delay chains

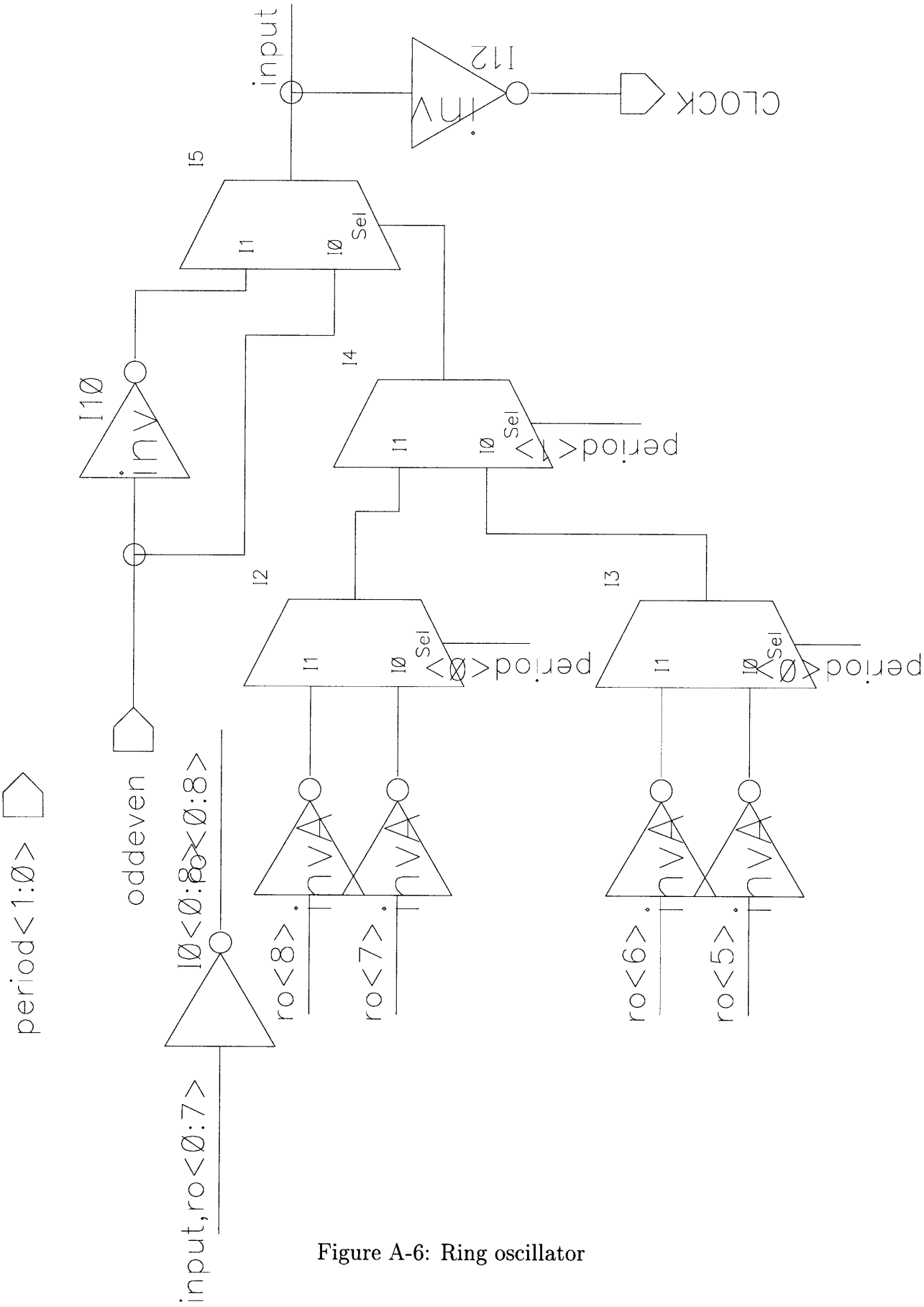

Figure A-6: Ring oscillator

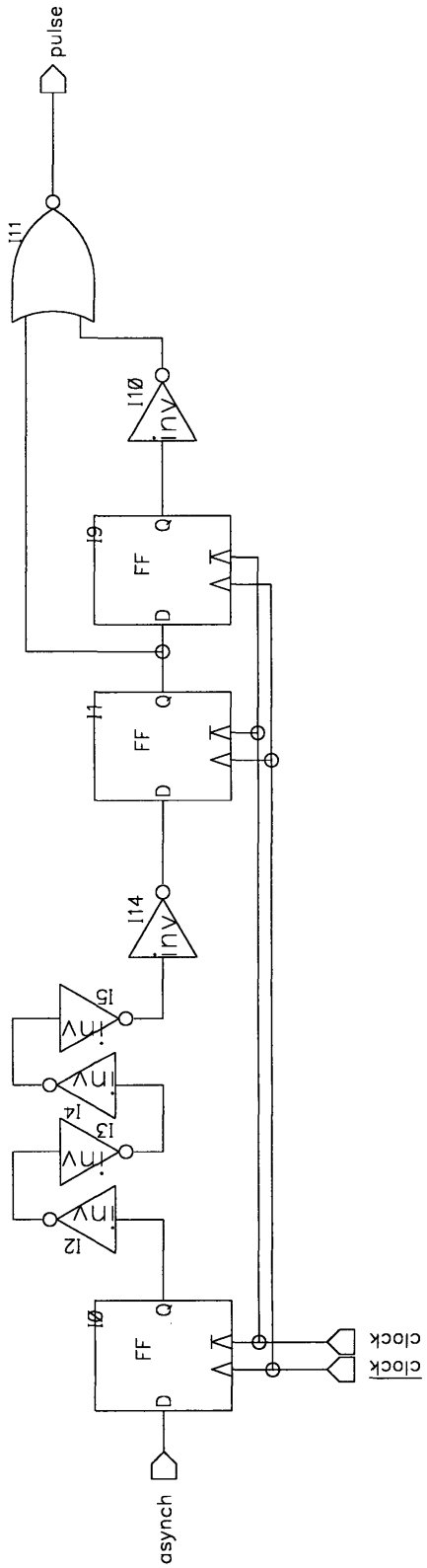

Figure A-7: Synchronization block