Appeared in "Proceeding of the 1993 Symposium on Integrated systems", Seattle, Washington, March 1993, MIT Press pp. 234-

Practical Implementation of Charge Recovering Asymptotically Zero Power CMOS

Saed G. Younis younis@ai.mit.edu
Thomas F. Knight, Jr. tk@ai.mit.edu
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
545 Technology Square
Cambridge, Massachusetts 02139, USA

Abstract

As clock and logic speeds increase, the power requirement of CMOS circuits are rapidly becoming a major concern in the design of personal information systems and large computers. In this paper we present a practical implementation of a new CMOS logic family, Charge Recovery Logic (CRL), with a power dissipation that falls with the square of the operating frequency, as opposed to the linear drop of conventional CMOS circuits. We show that such power saving is achieved with moderate increases in area and circuit complexity.

The technique relies on constructing an explicitly reversible pipelined logic gate, where the information necessary to recover the energy used to compute a value is provided by computing its logical inverse. Information necessary to uncompute the inverse is available from the subsequent inverse logic stage.

Depending on the Q of available inductors, we anticipate energy savings of over 99% per logic operation for logic which is clocked sufficiently slowly.

1 Introduction

In principle, a computing engine need not dissipate any energy, as shown in the work of Landauer, Bennett, and Feynman [8] [1] [3]. Although these authors approach the problem from different disciplines and use different physical as well as theoretical models, they all conclude that the transfer of energy through a dissipative medium dissipates arbitrarily small amounts of energy if this transfer is made sufficiently slowly. This should not come as a surprise: to eliminate the dissipation we have to make the potential applied to the medium zero, and from thermodynamics we know that the only way to transfer energy with zero potential is to allow the transfer to happen over an infinite amount of time.

Power dissipation in conventional CMOS primarily occurs during device

switching. One component of this dissipation is due to charging and discharging the gate capacitance through a conducting, but slightly resistive, device. We note here that it is not the charging or the discharging of the gate that is necessarily dissipative, but rather that a small time is allocated to perform these operations. In conventional CMOS, the time constant associated with charging the gate through a similar transistor is RC, where R is the ON resistance of the device and C its input capacitance. However, the cycle time can be, and usually is, orders of magnitude longer than RC. An obvious conclusion is that power consumption can be reduced by spreading the transitions over the whole cycle rather than "squeezing" it all inside one RC.

To successfully spread the transition over periods longer than RC, we forbid any device in our circuit from turning ON while a potential difference exists across it. Further, once the device is switched ON, the energy transfer through the device occurs in a controlled and gradual manner. While the latter is relatively easy to control through the use of inductors, the former leads to some interesting restrictions on the way we usually perform computations. We can always determine and control the potential on one side a CMOS device since it is usually connected to a power supply rail. The potential on the other side, however, depends solely on the result of the computation. To perform a non-dissipative transition of the output, we must know the state of the output *prior to and during* this output transition. Stated more clearly, to non-dissipatively reset the state of the output we must at all times have a copy of it. The only way out of this circle is to use *reversible logic*. It is this observation that is the core of a number of proposals to construct low power electronic computing engines. By low power, or non-dissipative, we mean that the energy per computational step can be made arbitrarily small by spreading the computation over a longer period.

Fredkin and Toffoli [4] demonstrated one realization of a low power universal gate using conservative logic. In conservative logic, the information content as well as the number of 1's and 0's are conserved throughout the computation. One property of a computation using conservative logic is the production of unwanted intermediate outputs. Unfortunately, discarding these outputs results in energy dissipation. Recycling it however, does not. It is this operation of recycling that requires the use of reversible logic [5]. The CMOS gate proposed by Fredkin and Toffoli could not be easily integrated, however, as it requires the use of inductors internal to the computational network. The sizes and numbers of these inductors are well beyond what can be easily accommodated on a silicon substrate.

Aware of the requirement to spread the energy transfer over a longer period of time, Seitz et al. [11] proposed a new reduced-power CMOS design style. The authors elected to use only N-Channel devices and therefore depended on bootstrapping action in order to eliminate the V_T voltage drop through their devices. The authors having successfully fabricated and op-

erated numerous circuits using this style still warned of the importance of device sizing to achieve enough bootstrapping for proper operation. More recently, Koller and Athas [7] proposed a form of dual rail logic that achieved adiabatic switching. Not using reversibility, they were unable to completely remove the power dissipation of their latches and commented that "to return the energy to the power supply without dissipating it, we must remember which output line we are charging. However, the only record we have of that is the very piece of information we are erasing". In addition, attempting to string multiple stages of this gate without intervening latches required a separate clock phase for each stage. The beauty of their implementation however is that their latch dissipation was about $\frac{1}{2}CV_T^2$ per bit, which is independent of V_{dd} and much less than the usual $\frac{1}{2}CV_{dd}^2$.

Recent and independent work by Hall [6] and Merkle [10] showed how to connect *Retractile Cascade* stages to eliminate the power dissipation in the latches. Merkle's proposal requires the use of intermediate voltage boosters made out of varactors and needs a relatively large number of independent clocks. In addition, the proposal calls for the explicit use of latches between the successive *Retractile Cascade* stages thus adversely affecting both throughput and latency.

In this paper we present a new technique for constructing non-dissipative CMOS circuits. We feel that this technique has a number of distinct advantages which warrant its use in future circuits. First, we require only two clocks and their inverses, for a total of four swinging rails, regardless of the number of stages. Second, each stage has an implicit latching action, and thus explicit latches between successive stages are not needed, simplifying pipelining and improving throughput and latency. Third, the technique, through proper sizing, allows the effective RC of the circuit as seen by each rail to be independent of the data values used in the computation. Since we use inductors to non-dissipatively move the charge, keeping RC constant is important for maintaining constant and data-independent rise and fall times. We note however, that correct functional operation of our gates does not depend on proper sizing. Fourth, we use CMOS pass gates and dual-rail logic throughout the circuit avoiding the need for bootstrapping, reversible varactor amplifiers, etc. Fifth, the topology and design rules of these circuits are identical to those of conventional CMOS except for a redundancy factor. Finally, we use reversible logic techniques to non-dissipatively reset each stage without ever reversing the direction of the pipeline.

The major disadvantage of our technique is the requirement for sixteen times as many devices as conventional CMOS. Area increases for large circuits may be far less, because of the similarity in the wiring patterns in duplicated logic. Four times as many signal wires are used, which may represent a more accurate estimate of the area ratios. The gates provide true and complement output forms of each logic value, eliminating the logical need for inverters, and partly compensating for the explosion in transistor count. We believe that for many applications, trading even a factor of six-

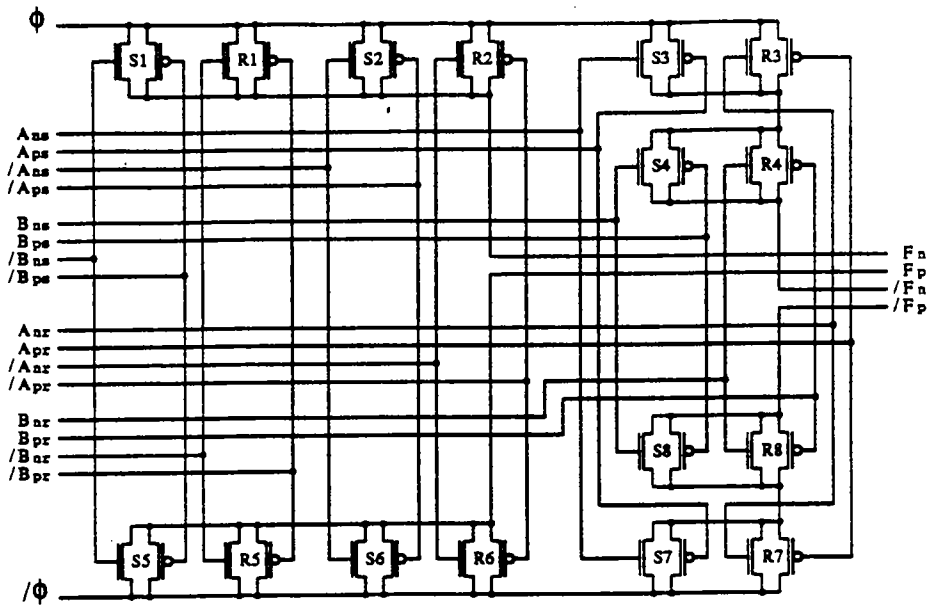


Figure 1: CRL 2-input NAND gate.

teen in silicon area is quite appealing for a power reduction of literally orders of magnitude.

2 Charge Recovery Gate

In this section we describe the topology and operation of CRL gates. We use the 2-input AND/NAND gate implementation shown in Figure 1 for our example. Because CRL uses pass gates instead of single transistors, we need to generate and distribute the *electrical* complement of each of our data lines in order to control the P-Channel side of the pass gates. We identify the lines intended to drive the N-Channel devices by the letter "n" in the subscript, and those for the P-Channel devices by the letter "p". Furthermore, our CRL circuit requires us to generate and distribute the *logical* complements of the data lines. We identify those in the figure by the forward slash character, /. We will clarify why we need these logical complements when we discuss the universality of this logic family.

From the above, we see that CRL uses four wires to represent each data line. An active data line A, representing the logic value 1, has $A_n = 1, A_p = 0, /A_n = 0, /A_p = 1$ for example. The reason for the apparent redundancy will become apparent shortly. In addition to the above, each CRL gate accepts two input groups. The first is the SET group, and we identify the lines belonging to this group by the letter "s" in the subscript. The second is the RESET group, and for it we use the letter "r" in the subscript.

CRL gates have two stable states. They are the SET state and the RESET state. While in the SET state, the output lines of a CRL gate are

valid and can be sampled by a subsequent gate. While in the RESET state, the output lines of a CRL gate must be forced to levels that will turn off any pass gates they are driving. Note that while *electrically* complementary lines are always at opposite levels, *logically* complementary lines are at opposite levels only when the gate driving them is in the SET state, thus the distinction between the two and the need for four wire signalling.

In going from the RESET state to the SET state, the gate uses the SET inputs to compute its valid output. At that time the gate assumes, and we require, that all the RESET inputs are idle. We define an idle input as one that is at a level that will turn off any pass gate it controls. In going from the SET state to the RESET state, the gate uses the RESET inputs to non-dissipatively restore itself to the RESET state.

In Figure 1, we have 16 pass gates labeled S1 thru S8 and R1 thru R8. The S gates are associated with the set inputs, and the R gates are associated with the reset inputs. Note that every SET pass gate is paralleled by a corresponding RESET pass gate. It is important that the set and reset pass gates be paralleled in the same order and with connections between intermediate pass gates in a series chain.

We are now ready to describe the operation of a CRL gate. We start with the gate in the RESET state. In the RESET state, the following is true in our example NAND gate:

- The top rail and all the nodes connected to it are at V_{ss}.
- All the sources and drains of the transistors in S1 thru S4 and R1 through R4 pass gates are at V_{ss}.
- All the outputs driving N-Channel devices are at V_{ss}.
- The bottom rail and all the nodes connected to it are at V_{dd}.
- All the sources and drains of the transistors in S5 thru S8 and R5 through R8 pass gates are at V_{dd}.
- All the outputs driving P-Channel devices are at V_{dd}.

We further assume that both the SET and RESET inputs are idle. We now wait until the SET inputs become valid. Note that since the voltage across every pass gate is now zero, turning the SET inputs active from their idle levels creates no dissipative transients. Recall that to go to the SET state, all the SET inputs must be valid and all the RESET inputs must be idle. Thus only the SET pass gates are active during this transition.

To SET the gate, we gradually swing the top rail from V_{ss} to V_{dd} and swing the bottom rail from V_{dd} to V_{ss}. Examining the circuit diagram and following the effects of these swings, we see that F_n will go high and F_p low if and only if the input data lines are such that $/(A \wedge B)$ is true. Otherwise, each will remain at its idle level while $/F_n$ swings high and $/F_p$ low. Therefore F, when active, computes $/(A \wedge B)$.

While in the SET state, we must guarantee that every pass gate that was off during the transition remains off for as long as the circuit is in the SET state. This is because a pass gate that was off during the SETting has a potential difference across it. On the other hand, turning off a pass gate that was on during the transition has no adverse effect.

We could trivially restore the circuit to its original state non-dissipatively by gradually returning the top and bottom rails to their RESET levels. Note that the SET inputs must remain valid while the circuit is reset in order to completely restore the state of the circuit, including the voltage levels of internal nodes. Unfortunately, it is not possible to build pipelined circuits using this simple scheme. A pipelined circuit accepts a new data value as soon as the first stage finishes processing the previous data point. However if every stage in the pipeline must wait until the subsequent stage resets before it can reset and accept a new value, then this "pipeline" can only accept a value after the previous value propagates to the end of the pipeline and after all the subsequent stages reset themselves in a reverse order. This circuit has the same throughput as a non-pipelined circuit.

We *can* however pipeline CRL gates if we do the resetting in the following manner. After SETting the circuit, we provide an identical copy of the SET inputs on the RESET lines. We will show how to generate these RESET inputs in Section 3. At the same time we allow the SET inputs to go idle, i.e. the previous pipeline stage can reset itself so as to accept new data inputs. Since every SET pass gate is paralleled by an identical RESET pass gate, we can now non-dissipatively restore the circuit by gradually restoring the top and bottom rails to their RESET levels. After RESETting the circuit, we allow the RESET inputs to go idle. Now, being in the RESET state, the circuit can accept new inputs.

The energy stored in a CRL circuit includes the energy stored in the internal nodes in addition to the energy stored on the output lines. By providing exact duplicates of the set inputs on the reset lines, and by connecting reset and set pass gates in parallel everywhere, we guarantee that the energy on these internal nodes is recovered.

Generating and distributing the logical complement of every line doubles both the wire and device counts. The device count doubles because of the complementary networks that are needed to generate them. In the circuit of Figure 1, S1, S2, R1, R2, S5, S6, R5 and R6 belong to the networks generating true outputs. The remaining pass gates belong to the networks generating the logical complements. Just as in conventional CMOS gate where the P-Channel network is the complement of the N-Channel network, in CRL gates, the network that generates a complementary output is the complement of the network that generates the true output. We pay this price of generating and distributing the logical complement to retain a crucial property of any logic family, *universality*. Since when a line is idle, the pass gates it controls are off, a logical 0 input cannot produce a logical 1 output. We need the complementary outputs in order for CRL family

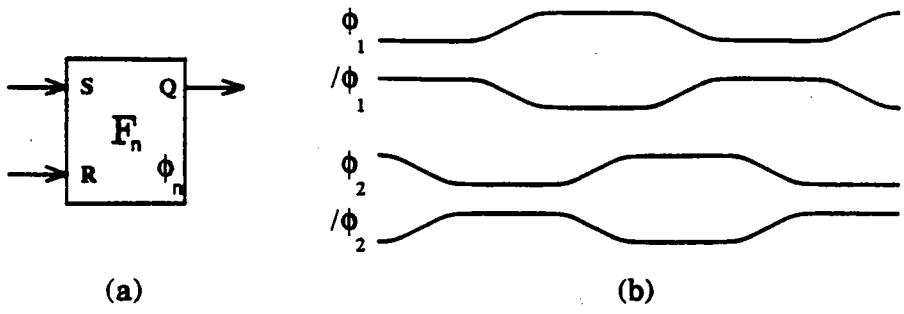


Figure 2: (a) Abstraction box. (b) Timing of the four clock rails.

to support negation. By demonstrating the implementation of a 2-input AND/NAND gate, we equivalently have shown that CRL is universal. It should be obvious how we can build any and-or logic function based on the above techniques in a way similar to conventional CMOS.

So far, our CRL gate requires eight times as many devices as conventional CMOS as illustrated by the 2-input AND/NAND gate example. For circuits that require complementary outputs, such as address decoders, the redundancy factor may be somewhat less.

3 Network Connection

In this section we show how to connect CRL gates, or stages, in a non-dissipative pipeline. The main purpose of this method of interconnection is to provide the RESET inputs to each gate at the correct time. We build the pipeline out of copies of an abstraction box shown in Figure 2a.

We think of this box as containing a parallel set of CRL gates performing any logical function of an arbitrary number of inputs. Symbolically, the output of the box represents a bundle containing the outputs of the CRL gates internal to the box. The box has two input branches. One is the SET branch and identifies a bundle containing all the SET inputs of the gates internal to the box, the other is the RESET branch and identifies a bundle containing all the RESET inputs. The function computed by the box is identified by the letter in the center of the box. Finally, indicated at the lower corner of the box is the clock phase used to control all the CRL gates internal to that box. A clock of ϕ_1 indicates that the top rail is connected to ϕ_1 and the bottom rail to $/\phi_1$, while a clock of $/\phi_1$ indicates that the top rail is connected to $/\phi_1$ and the bottom rail to ϕ_1.

Using this abstraction, Figure 3 illustrates how CRL gates are connected to produce a non-dissipative pipeline. The timing of the four clock lines is shown in Figure 2b. Note that the box with a function F^{-1} performs the inverse operation of the box with a function F. To SET a box, all the SET inputs must be valid and stable and all the RESET inputs must be idle, i.e.

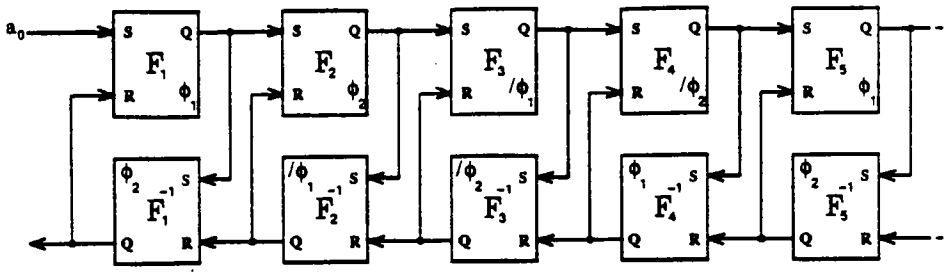


Figure 3: Non-dissipative multi-stage pipeline connection.

they come from a box that is currently in RESET, so that all the RESET pass gates are OFF. With these inputs, swinging the clock rails of the box away from their rest level will SET the box. To RESET the box, the rails are returned to their rest levels while the SET inputs are idle and the RESET inputs are active and stable.

To follow the operation of the circuit we start with ϕ_1 and ϕ_2 at their rest state and assume that the pipeline has been operating for some time. We follow the propagation of the input a_0 only, even though other parallel activity is going on. From the states of the clocks we see that box F_1 is RESET and its RESET inputs are idle as well. Swinging ϕ_1 SETs F_1 and computes $F_1(a_0)$. Swinging ϕ_2 now SETs F_2 and F_1^{-1} and produces $F_2(F_1(a_0))$ and $F_1^{-1}(F_1(a_0)) = a_0$ respectively. Now swinging ϕ_1 to its rest level RESETs F_1, produces $F_3(F_2(F_1(a_0)))$ and $F_2^{-1}(F_2(F_1(a_0))) = F_1(a_0)$. The circuit is now ready to safely RESET boxes F_2 and F_1^{-1}. One can see that we can continuously drive a new input into the network every ϕ_1 and successfully operate the pipeline in a non-dissipative fashion. In addition, this pipeline can have any arbitrary number of stages and still be driven entirely by ϕ_1, ϕ_2 and their inverses only.

There remains one problem however. At the end of the pipeline the RESET input to F_5^{-1} is not available and hence resetting this box is dissipative. Furthermore, it could not be generated, as this is the place where reversibility is broken. We can however, restore reversibility here through brute force by connecting to the end of this pipeline a mirror, and an inverse, image of itself. The missing input at the end of this extended pipeline that is needed to reverse the last inverse box is now simply a_0. With this topology, we can proceed without any dissipation by continually supplying delayed copies of the input to the pipeline at the inverse input on the far right. The technique of connecting an inverse network to the forward network was previously used in [5] and [3] to eliminate dissipation through recycling the intermediate *garbage* that results in conservative logic.

Admittedly, the above solution is more of theoretical than practical interest. If reversibility needs to be broken, that is, when information loss cannot be avoided, then some dissipation will occur for every lost bit of information. For these situations, we can reduce the dissipation by ending the pipeline

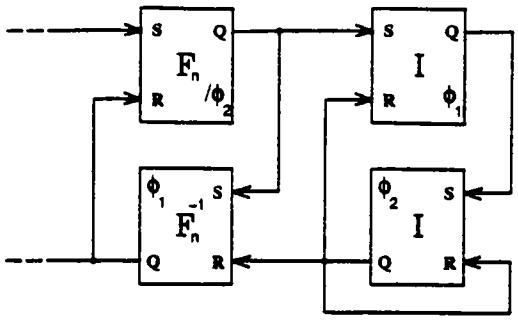


Figure 4: Last pipeline stage connection for dissipation reduction.

with two identity boxes, $I(a) = a$, and use the output of the lower identity box to reset itself as shown in Figure 4. Closer examination shows that the dissipation is $\frac{1}{2}CV_T^2$ per bit per cycle as opposed to $\frac{1}{2}CV_{dd}^2$ for conventional gates. Since the output of an identity box is the same as the input, the resetting swing proceeds normally until the output levels are insufficient to keep the appropriate pass gates on. Because of this, some internal nodes will have a potential that is one V_T away from their reset levels. The next input to the gate will short this potential difference resulting in $\frac{1}{2}CV_T^2$ dissipation per bit. Note that that we only pay this penalty at the last stage of a long pipeline.

4 Reversibility

The reason for choosing a NAND gate for the implementation example was to demonstrate the universality of CRL . Obviously, the NAND function is not reversible. To be reversible, a function must minimally have as many output as it has inputs. In addition, a non-conservative function generally requires more outputs than inputs to be reversible. If in addition to generating $/(A \wedge B)$ we pass thru a copy of A and a copy of B, then we would have enough information to make this augmented NAND gate reversible. So long as the information entropy of the system is kept constant, making every functional block of the system reversible is always possible and usually simple. However, an operation resulting in an increase in the informational entropy of the system is fundamentally irreversible and hence dissipative. With infinite storage we can always maintain enough information history to reconstruct the past regardless of the nature of the computation. Fortunately, we can be clever about trading off a little dissipation at critical places in the computation where reversibility would be complex or costly and use the system's *finite* resources to maintain reversibility and asymptotic zero dissipation elsewhere. CRL in essence provides the link between power dissipation and information content.

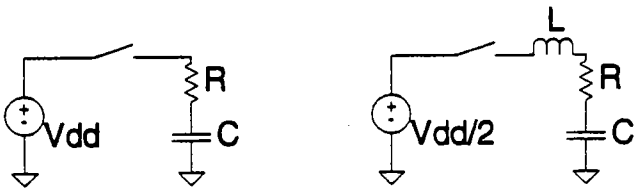


Figure 5: Conventional and non-dissipative circuit analysis models.

5 Power-Delay Product

The power-delay product is used as a figure of merit to compare logic families. In this section we compare the performance of a conventional CMOS gate with that of our gate on the basis of power dissipation per period. We use the circuits shown in Figure 5 as our models. These models are based on a lumped element approximation where the R's and C's are the effective R and C as seen from the supply terminals of the chip. For appropriately sized circuits we can assume that the gate voltages have roughly the same rise time and therefore we can approximate R by R_{device}/N and C by $C_{device} \times N$ for an N gate chip. Note that R and C depend only on N and the fabrication process technology. In conventional CMOS, to go through a single period of operation, we discharge C from an initial voltage of V_{dd} to ground and then charge it back to V_{dd}. For our CRL gate to go through a single period we use inductors connected to $V_{dd}/2$ on one side to swing the rails to the opposite polarity at the appropriate time. To swing a rail that rests at V_{dd} we

1. disconnect the rail from the positive supply and connect it to the inductor,
2. we keep the inductor connected until the current reaches zero signaling a complete polarity inversion,
3. we disconnect the rail from the inductor and connect it to the negative rail,
4. and finally we reverse the above steps to go back to the positive state.

The inductor in the above circuit acts as an electrical "flywheel" that inverts the polarity of internal nodes connected to the rail. For now, assume the switch connecting the rail to the inductor is external to the chip. We will revisit this decision in a later section.

As is widely known, the internal energy dissipation of conventional CMOS circuits is attributable to three major components. The first is due to the static leakage current, which we will assume to be negligible. The second is due to the transient current associated with charging and discharging the gate capacitance C through the ON resistance R and is equal to $C V_{dd}^2$ per

period for a rail-to-rail voltage equal to V_{dd}. The third is due to the switching current which is caused by both N-Channel and P-Channel devices being simultaneously ON for a brief time during a swing of $V_{dd} \geq 2V_T$ [2]. We can approximate this switching dissipation by

$$V_{dd} \times \frac{1}{2}\frac{V_{dd}}{2R} \times \frac{V_{dd} - 2V_T}{V_{dd}} \times 2RC$$

which for a typical $V_{dd} = 4V_T$ simplifies to $\frac{1}{4}CV_{dd}^2$. As V_{dd} drops below $2V_T$ this switching dissipation becomes negligible. Unfortunately, the transfer curve begins to exhibit hysteresis thus limiting the utility of the gate at these power supply voltages.

Since the switching dissipation has a form similar to the transient dissipation, and is usually smaller in magnitude, we concentrate on the dissipation due to the transient dissipation associated with charging the gate capacitances of the devices, and approximate the total dissipation per period of conventional CMOS, E_{CMOS}, as

$$E_{CMOS} = CV_{dd}^2$$

We now examine the dissipation of our proposed circuit. To simplify the algebra, we let $V_{dd} = +V_0$ and $V_{ss} = -V_0$ so that V_0 is equal to half the rail-to-rail voltage V_{dd}. Our R,L,C circuit is described by

$$\frac{d^2V_C}{dt^2} + 2\alpha\frac{dV_C}{dt} + \frac{V_C}{{\omega_0}^2} = 0$$

where V_C is the capacitor voltage, $\alpha = R/2L$, and $\omega_0 = 1/\sqrt{LC}$. For the solution to oscillate, the circuit must be *underdamped*, requiring that

$$2\sqrt{LC} > RC$$

and we find that the frequency of oscillation, ω_d, is given by

$$\omega_d = \sqrt{{\omega_0}^2 - \alpha^2}$$

Since R and C are fixed for a given chip, we can only adjust L to affect ω_d. Examining the formula for ω_d we discover that ω_d steadily increases as L decreases up to a maximum, ω_{dmax}, and then sharply decreases as the circuit approaches the *critically damped* point. We find that

$$\omega_{dmax} = \frac{1}{RC} = \frac{1}{R_{device}C_{device}}$$

and the smallest inductance we would ever need, L_{min}, is found by

$$L_{min} = R^2C/2 = R_{device}^2C_{device}/2N$$

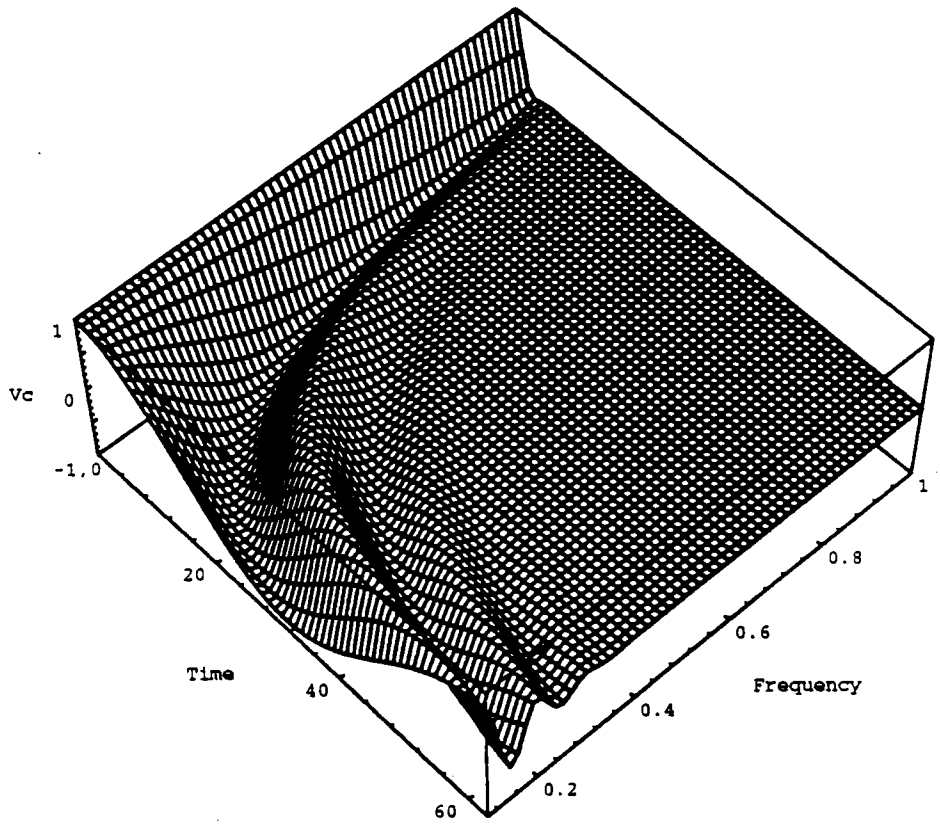


Figure 6: Plot of $V_c(t, \omega_d)$ showing the effect of fast rise and fall times on energy loss.

For $V_C(0) = V_0$ and $i_C(0) = 0$ we find

$$V_C(t) = V_0 e^{-\alpha t}(cos\omega_d t + \frac{\alpha}{\omega_d} sin\omega_d t) \tag{1}$$

and

$$i_C(t) = V_0 C \frac{{\omega_0}^2}{\omega_d} e^{-\alpha t} sin\omega_d t \tag{2}$$

Figure 6 shows a plot of normalized V_c as function of time and normalized ω_d. Initially $V_C(0) = 1$. With ω_d close to ω_{dmax} the voltage drops rapidly, dissipating most of the capacitor energy on the way down. The reclaimed energy is insufficient to fully charge the capacitor back to the negative rail. As ω_d moves away from ω_{dmax} we in effect spread each oscillation over a longer period of time. We see from the plot that the capacitor retains most of its energy since V_C comes close to the bottom rail. With ω_d only an order of magnitude lower than the maximum, the capacitor recovers most of its energy.

The power dissipated in a single rail-to-rail swing for our gate consists of two components. The first is the power dissipated in R during the swing,

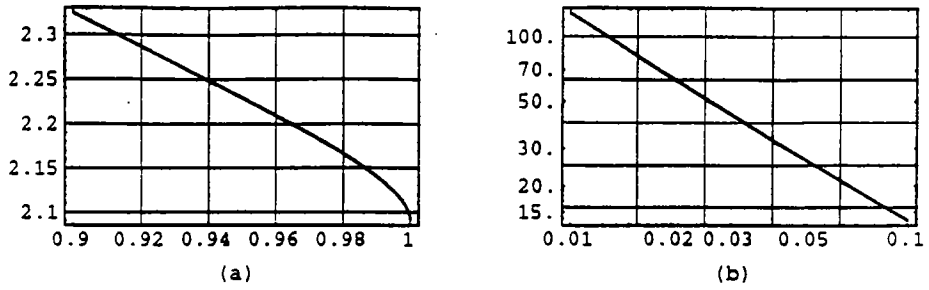


Figure 7: (a) Linear plot of F_{saving} factor vs. ω_d. (b) Log-Log plot of F_{saving} factor vs. ω_d. In both plots ω_d is normalized to ω_{dmax}.

E_s, which results in a final voltage that is lower than $|V_0|$. Using equation 1 we find that

$$E_s = \frac{1}{2} C\ V_0^2\ (1 - e^{\frac{-2\pi a}{\omega_d}})$$

The second is the energy lost in R while charging C to the rail to compensate for the lower peak voltage due to E_s. We will do this by simply connecting the line to the rail and losing some energy in R, E_c, during this process. We find that

$$E_c = \frac{1}{2} C\ V_0^2\ (1 - e^{\frac{-\pi a}{\omega_d}})^2$$

adding these two terms and multiplying by two to allow for both directions of swing, we find that

$$E_{loss} = 2C\ V_0^2\ (1 - e^{\frac{-\pi a}{\omega_d}}) \qquad (3)$$

per period.

Using Equation 3, we compute the ratio of the power consumption per period of conventional CMOS to that of our gate, F_{saving}, as

$$F_{saving} = \frac{2}{(1 - e^{\frac{-\pi a}{\omega_d}})}$$

Figure 7a shows a Linear plot of F_{saving} near ω_{dmax}. In this region, the circuit's performance is close to conventional CMOS but improves rapidly with lower ω_d. As we get away from ω_{dmax}, the graph attains a nearly constant slope as shown in Figure 7b. Note that we only need to slow down a little in order to see substantial power saving. For $R_{device} = 10\text{K}\Omega$ and $C_{device} = 0.1\text{pF}$, $\omega_{dmax} = 1\text{GHz}$. Operating at a speed of 40 MHz we will consume 33 times less power and at a speed of 4 MHz we achieve an impressive 319 times power saving. We emphasize that these numbers reflect the power saving advantage per period and are therefore in addition to the linear power drop as a function of frequency found in conventional CMOS.

6 Dynamic Considerations and Nonlinearities

The above analysis assumes that we can, in theory, lump the gate capacitances of MOS devices into one equivalent linear capacitance. In practice however, we need to be more careful. Each rail in CRL feeds a number of branches. Ideally, the effective RC time constant of each branch as seen by the rail is data independent and equal to the RC time constant of the entire rail circuit. A branch with a longer time constant would lag behind during the transition. This would create a potential difference across pass gates that are switched on, leading to dissipation. The effect is minimized by the symmetry of CRL . Because of the existence of the true and complement networks for every output line, a swinging rail is always connected to one and only one output line. Therefore, regardless of the output level, the rail will always drive the same output capacitance. The only difference in RC comes from the fact that the true and complement networks are not identical and as such could contribute different RC's depending on the data. Properly sizing the devices so that the two networks exhibit the same time constant independent of the state of the inputs will eliminate this problem.

Another point of consideration is the nonlinearity of the capacitances of MOS devices. An enhancement mode MOS device has a higher gate capacitance while in inversion, i.e. conducting, than when it is off. At the beginning of a SETting swing, all the outputs are at idle and all the devices driven by these outputs are off. At the completion of a SET, devices controlled by a swinging output are on. For this reason, a rail that is SETting a gate sees lower effective capacitance at the start of the swing than at the end. With the inductor anchored to $V_{dd}/2$, the rail will not reach the opposite voltage at the end of a swing. This leads to dissipation when the rail is connected to V_{dd} or V_{ss} after completing the swing. Fortunately, a rail SETting a gate is at the same time RESETting another. We feel that as the number of gates connected to a rail increases in a balanced way, the adverse effects of this nonlinearity is minimized. Intuitively, the effective RC time constant is now equal to $R \times C_{average}$.

In addition to the above, there is also the effect of capacitive coupling. Take an output wire that carries a logical true. The devices that it controls in a subsequent gates are on. Because of the gate-to-channel capacitance, when the subsequent gates SET, the varying voltage in the channels of turned on devices capacitively couples to the gate. This dumps, or extracts, charge from the output wire of the previous stage. Again, the effect is minimized due to symmetry. Since each output drives identical devices that are capacitively coupled to the top and bottom rail swinging in opposite directions, this capacitive coupling is almost entirely eliminated by the symmetry. We say almost because due to the capacitance nonlinearity, the symmetry will cancel the coupling when integrated over the entire swing and not instantaneously.

We want to stress here that while minimizing the effects of the above phenomenon improves the power saving factor of CRL circuits, none of

the above effects, even when extreme, jeopardizes the logical functionality of CRL circuits. HSpice simulations of the 2-input NAND gate, as well as simulations of other CRL gates and circuits, have demonstrated proper operation of the CRL circuit in the presence of these effects. This is important in simplifying the design of CRL logic. If a designer incorrectly sizes a branch in a CRL circuit, the worst he can expect is higher power dissipation in that part of the circuit and not a disfunctional chip.

7 Practical Issues

We have presented a logic family in which each transistor internal to the logic is turned on only when there is zero voltage across it, thus eliminating energy dissipation in the switch. There remain, however, a number of practical implementation issues.

First, while we have minimized the use of external inductors by requiring only four external clock rails, there remains potential dissipation associated with the switch in series with each inductor. This switch is now the only component with significant voltage across it when it is turned on. This is where we expect to see significant power loss. Yet, we are switching the transistor at a time when we can guarantee, as a result of the series inductance, that there is near zero current flow. If we can fully turn on the transistor prior to significant current build up, then dissipation can be minimized.

Turning this switch transistor on requires charging and discharging its gate capacitance. If we do this with a dissipative switching operation, then much of our potential power savings will be thrown away here.

However, we can recursively apply the power recovery techniques discussed above to recover the energy stored in the switch transistor gate. Presumably, applying a single level of such recursive power recovery will be adequate for most applications.

A second difficulty in fabricating circuits with high power recovery is the Q of the inductor. While our derivation assumed an infinite inductor Q, realistic high frequency inductors have Q's limited to the range of 100-200, placing an upper limit on the achieved power recovery.

In some applications, the use of high temperature superconducting high Q inductors, in combination with liquid nitrogen cooled CMOS might be an attractive combination. Achievable Q's with superconducting technology are in the several hundred thousands, and the major limitation might become leakage currents in CMOS devices. These dark currents will also be drastically reduced by low temperature operation. Lowering the temperature also increase the carrier mobility. At liquid nitrogen temperatures, the mobility of carriers in MOS devices increases 4 times under low field conditions and 1.7 times under high field conditions. Operating under both high and low field conditions, the measured mobility increase in conventional CMOS

averages to about 2.5 times [9]. In contrast, CRL operation is by design limited to low field operation and would therefore retain the full benefit of 4 times mobility increase at LNT.

8 Reliability

The reliability of CRL circuits should be excellent. First, the low dissipation of CRL circuits ensure operation at near ambient temperatures with minimal attention to cooling. Second, during conventional CMOS switching, the circuit temperature can rise above the average temperature for a short time. Since the failure rate is related to a high power of temperature, the overall failure rate is typically higher than that predicted by the average operating temperature. As CRL circuits disallow sudden transients, the failure rate of CRL devices will be lower than that of conventional CMOS operating at the same temperature. Third, the elimination of transistor conduction in the presence of high drain to source voltages removes hot electron trapping as a failure mechanism. This may allow finer transistor geometry, higher performance, and process simplification by elimination of LDD and sidewall spacer techniques used to control hot electron behavior in conventional technologies. Finally, the peak currents in CRL are minimized thus further reducing other failure effect such as metal migration.

9 Conclusion

We have demonstrated a set of techniques for constructing a reversible, energy efficient family of computing devices, with theoretically near-zero dissipation. Key ideas include the reduction in the number of external swinging rails through the use of our phased reverse computation technique and the paralleling of all forward computing pass gates with a second pass gate driven by the recomputed logic values.

An analysis of the available energy savings demonstrates the asymptotically zero dissipation available. We presented a discussion of practical issues and considerations in the construction of these circuits.

Acknowledgments

This research is supported in part by the Defense Advanced Research Projects Agency under contract N00014-91-J-1698. We acknowledge useful discussions with Carl Feynman, Norman Margolis, Tom Simon, and Tom Toffoli.

References

[1] Bennett, C., "The Thermodynamics of Computation – a Review", *International Journal of Theoretical Physics*, Vol. 21, No. 12, 1982, pages 905-940.

[2] Calebotta, S., "CMOS, the Ideal Logic Family", *National Semiconductor CMOS Databook,* Rev. 1, AN-77, pp. 2-3, 1988.

[3] Feynman, R., "Quantum Mechanical Computers", *Foundations of Physics,* Vol. 16, No. 6, 1986.

[4] Fredkin, E., and Toffoli, T. (1978),"Design Principles for Achieving High-performance Submicron Digital Technologies," Proposal to DARPA, MIT Laboratory for Computer Science.

[5] Fredkin, E., and Toffoli, T., "Conservative Logic", *International Journal of Theoretical Physics*, Vol. 21, Nos. 3/4, 1982, pages 219-253.

[6] Hall, J. S., " An Electroid Switching Model for Reversible Computer Architectures," in *Proceedings of Physics of Computation Workshop,* Dallas Texas, October 1992.

[7] Koller, J. G., and Athas, W. C., "Adiabatic Switching, Low Energy Computing, and the Physics of Storing and Erasing Information," in *Proceedings of Physics of Computation Workshop,* Dallas Texas, October 1992.

[8] Landauer, R., "Uncertainty Principle and Minimal Energy Dissipation in a Computer", *International Journal of Theoretical Physics* , Vol. 21, Nos. 3/4, 1982, pages 283-297.

[9] Laramée, J., Auburn, M.J., and Cheeke J.D.N., "Behavior of CMOS Inverters at Cryogenic Temperatures," *Solid-State Electronics*, vol. 28, no. 5, pp. 453-456, May 1985.

[10] Merkle, R. C., "Reversible Electronic Logic Using Switches", Submitted to *Nanotechnology*, 1992.

[11] Seitz, Charles L. *et al.*, "Hot-Clock n*MOS*," in *Proceedings of the 1985 Chapel Hill Conference on VLSI,* Computer Science Press, 1985.

Addendum

10 New Charge Recovery Logic Circuit

The circuit described in section 2 of the paper requires 8 times as many devices as conventional CMOS circuits. The following is a description of alternate implementations that require 2-4 times as many devices only. The tradeoff is that the following circuits require more swinging rails, 6-8 rails, than what the previous implementation needed, 4 rails. To Fully understand the following description of the improved circuit we suggest that one read the paper describing the previous circuit first.

The main idea behind these new circuits is that they have two sections. A front-end section that has only N-Channel devices, and a back-end section that uses pass gates made out of N-Channel and P-Channel devices. The new circuits require two different rails that are not the complement of one another. The first rail, we call the slow rail, controls the front-end section of the circuit. The second, we call the fast rail, controls the back-end of the circuit. Figure 1 shows an implementation of a NAND gate using this new techniques and indicates the front-end and back-end sections. In the front-end section, all SET N-Channel devices, identified by the letter "S" in the figure are parallel by RESET N-Channel devices, identified by the letter "R". The pass gates of the back-end section do not have RESET pass gates. While in the RESET state, all the outputs and internal nodes of the front-end section are at V_{ss}. The slow rail is at V_{ss} as well. In addition, all the outputs and internal nodes of the back-end section are at V_{ss}. The fast rail would be at V_{ss} as well. We assume that all the SET and RESET inputs are at V_{ss}. The circuit is now ready to accept new input on its SET lines. After the SET inputs become valid and stable, we gradually swing the slow rail from V_{ss} to V_{dd}. At the end of the swing, some outputs of the front-end section would remain at V_{ss} while some would swing to $(V_{dd} - V_T)$ depending on the input value and the implemented function. Note that we generate the true and complement of every signal in the front-end section so that we could drive both sides of the pass gates in the back-end section. One of the main purposes of the back-end is to regenerate the rail-to-rail logic levels at the output of the gate that could not be generated by the N-Channel only section. After the front-end SET's, we swing the fast rail from V_{ss} to V_{dd}. Depending on the computed result, we could have the pass gates of the back-end set ON or OFF. Those that are set on by the front-end, having their P-Channel side at V_{ss} and their N-Channel side at $V_{dd} - V_T$ will swing the outputs they are driving to V_{dd}. The Off pass gate would have their P-Channel at $V_{dd} - V_T$ and their N-Channel side at V_{ss}. While the N-channel side is OFF and will remain OFF during the entire swing of the fast rail, the P-Channel could start conducting just before the end of the swing. However, assuming that the threshold voltages of the P-channel and

the N-Channel devices are roughly equal, except for the sign, the channel-to-gate capacitive coupling of the P-channel devices will raise the voltage at their gate during the low to high swing of the fast rail thus insuring that they will remain off. Note that the bootstrapping here is used to shut off devices and not to recover the V_T drop of the front-end. For his reason, the minimal of coupling would still result in keeping the device off and would lead to proper operation. Note that the outputs of the back-end are now rail-to-rail and could drive the front-end of a subsequent stage without V_T degradation.

To RESET the circuit, we provide wait until the RESET inputs are active and the SET inputs go idle. First we swing the fast rail back to V_{ss} and then reset the circuit by returning the slow rail to V_{ss}. We need to reset the fast rail before the slow rail because the back-end does not have any RESET devices in parallel with the SET devices. The fact that the fast rail must SET and RESET itself while the slow rail is at the SET level forces it to have a much narrower duty cycle and hence the name *fast* rail. Because of the different duty cycles, we now need a total of eight rail, instead of 4, to run our circuits. The timing of these rails is shown in Figure 2.

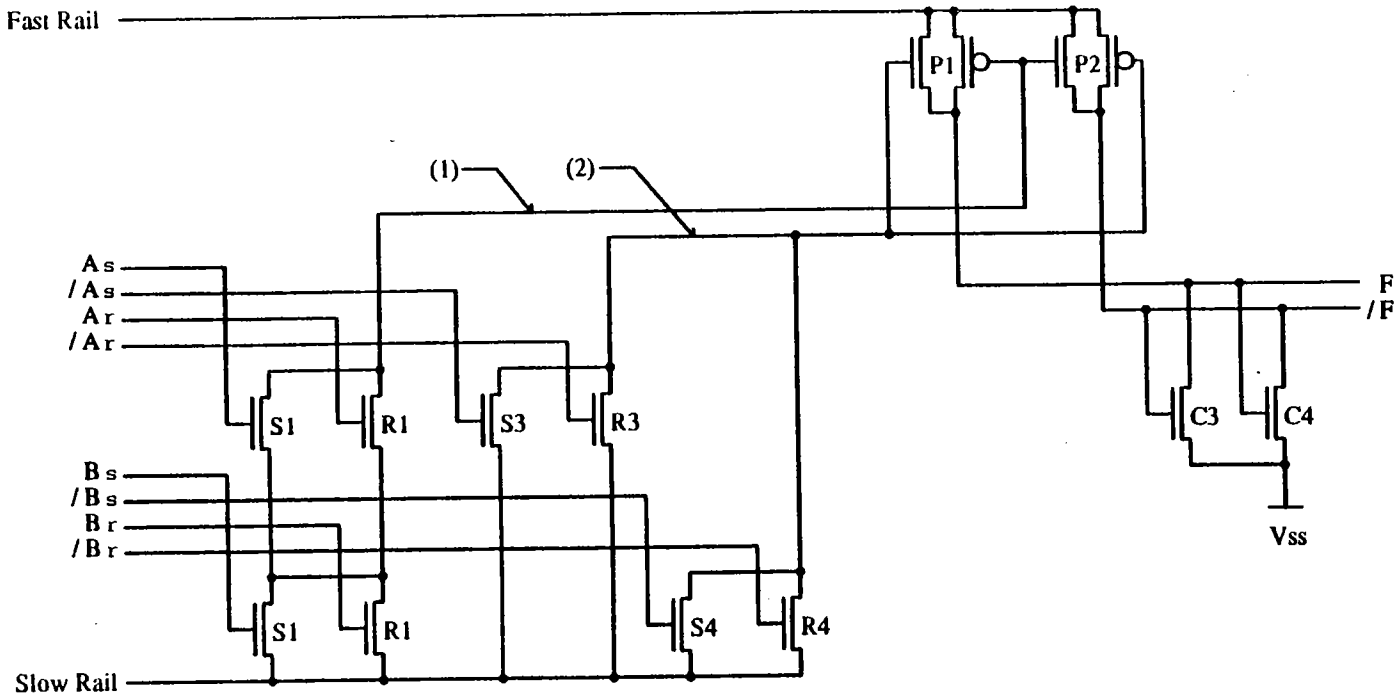
In addition to the devices that are used for computing, the new circuit has N-Channel cross coupled pairs that are tied to the outputs of front and back sections of the circuit. These devices are used to hold the voltage of the node that does not swing to the rest rail and hence to maintain proper operation in the presence of dark currents.

So far we have assumed that the front-end section RESET's to V_{ss} and SET's to V_{dd}. However, by modifying the cross coupled devices only as shown in Figure 3, we can have it so that the front-end SET's to V_{ss} and RESET's to V_{dd}. Having done that, we could drive the gates that was supposed to be driven by the complement of a slow clock by the true clock itself. This eliminates the need for the complements of the slow clock and reduces the needed swinging rails from 8 to 6.

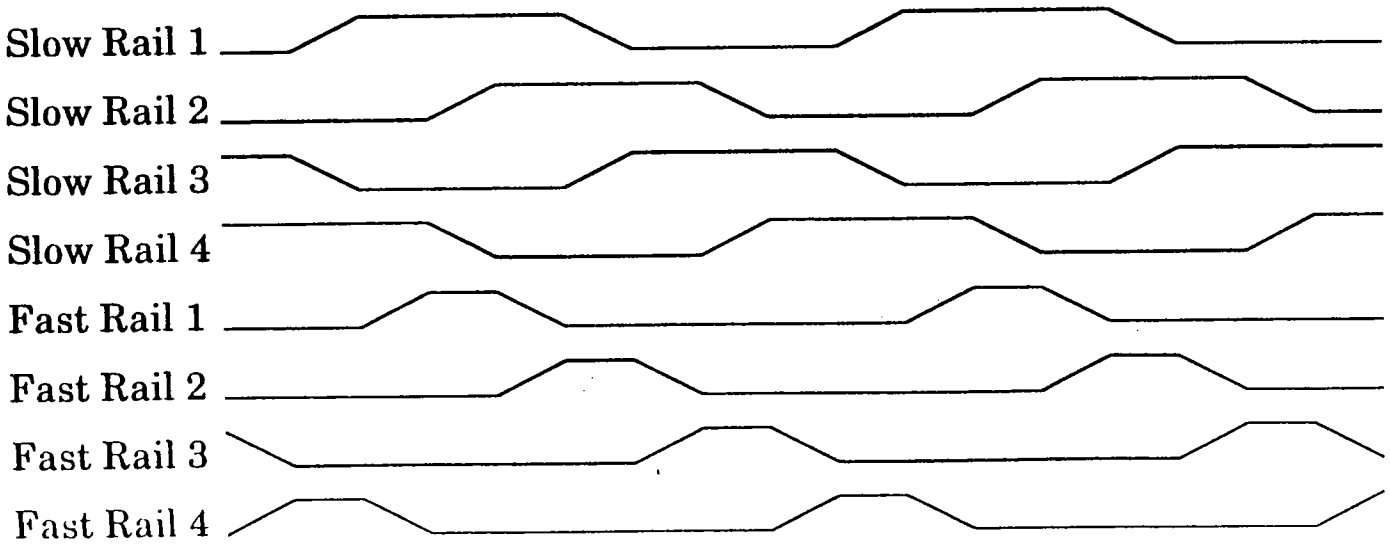
The circuit described above, could be connected in the same manner described in section 3 of the previous paper in order to yield a pipelined low power circuit.

In addition to the reduction of the number of devices needed, the above circuit has the added advantage of simplifying reversibility. We can stack the pass gates of the back-end to do computation just as the front-end.Since it is the function of the whole gate, consisting of the two section, that must be reversible, and not the function of the subsections, we can embed non-reversible functions in the front-end section of our gate and not worry about it so long as the function of the whole gate is reversible. For example, we build an adder gate that is easily reversible from AND and OR gates that are not easily reversible but hidden within the bigger reversible adder block.

Schematic diagram of a 2–Input AND/NAND Gate

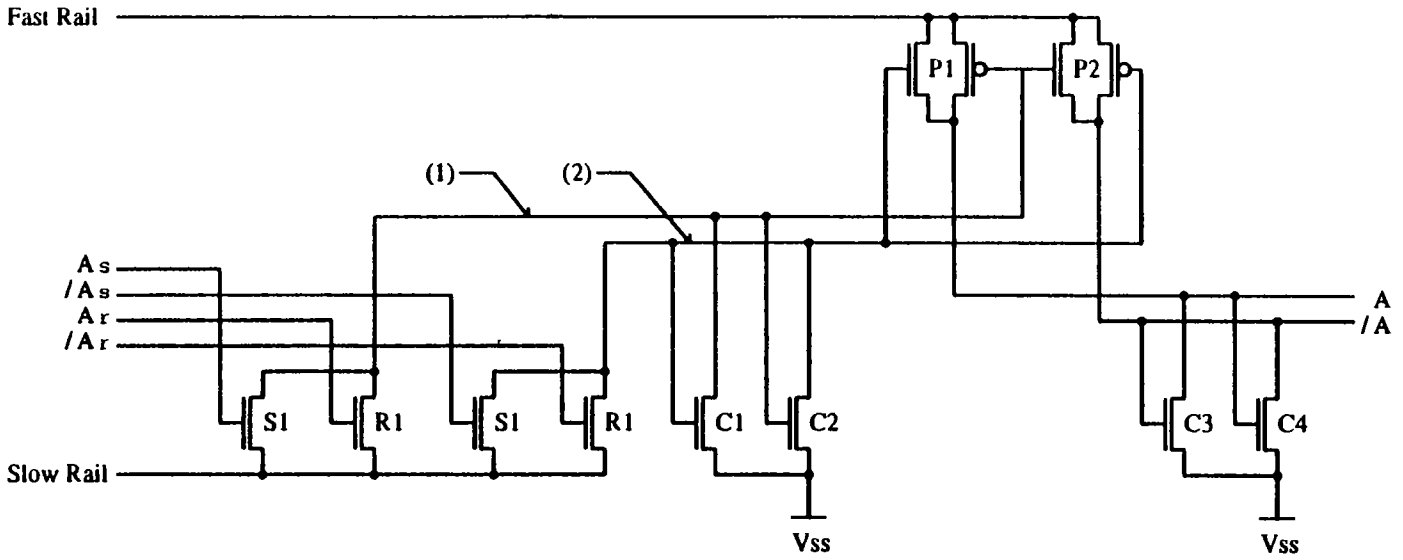


Rails Timing Diagrams

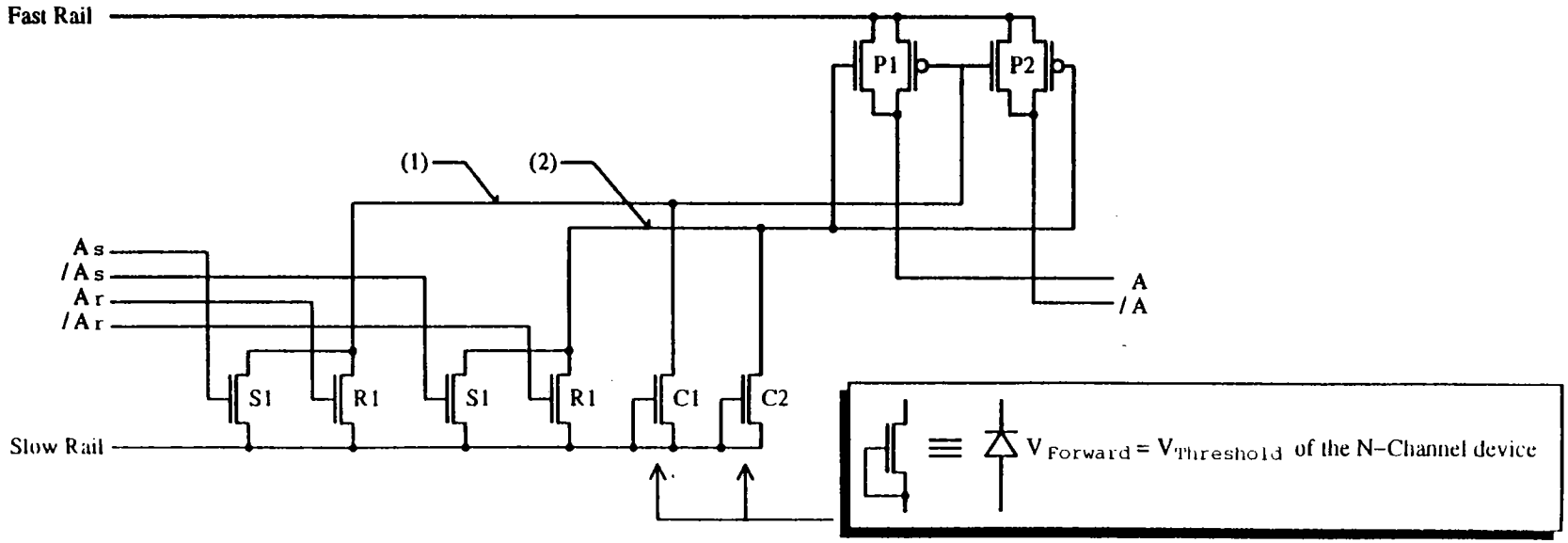


Schematic diagrams of a 1 bit CRL register in the new technology. The two variations are included below. In both versions the Fast Rail SET's to 5V and rests at 0V. The versions differ in where the Slow rail sets.

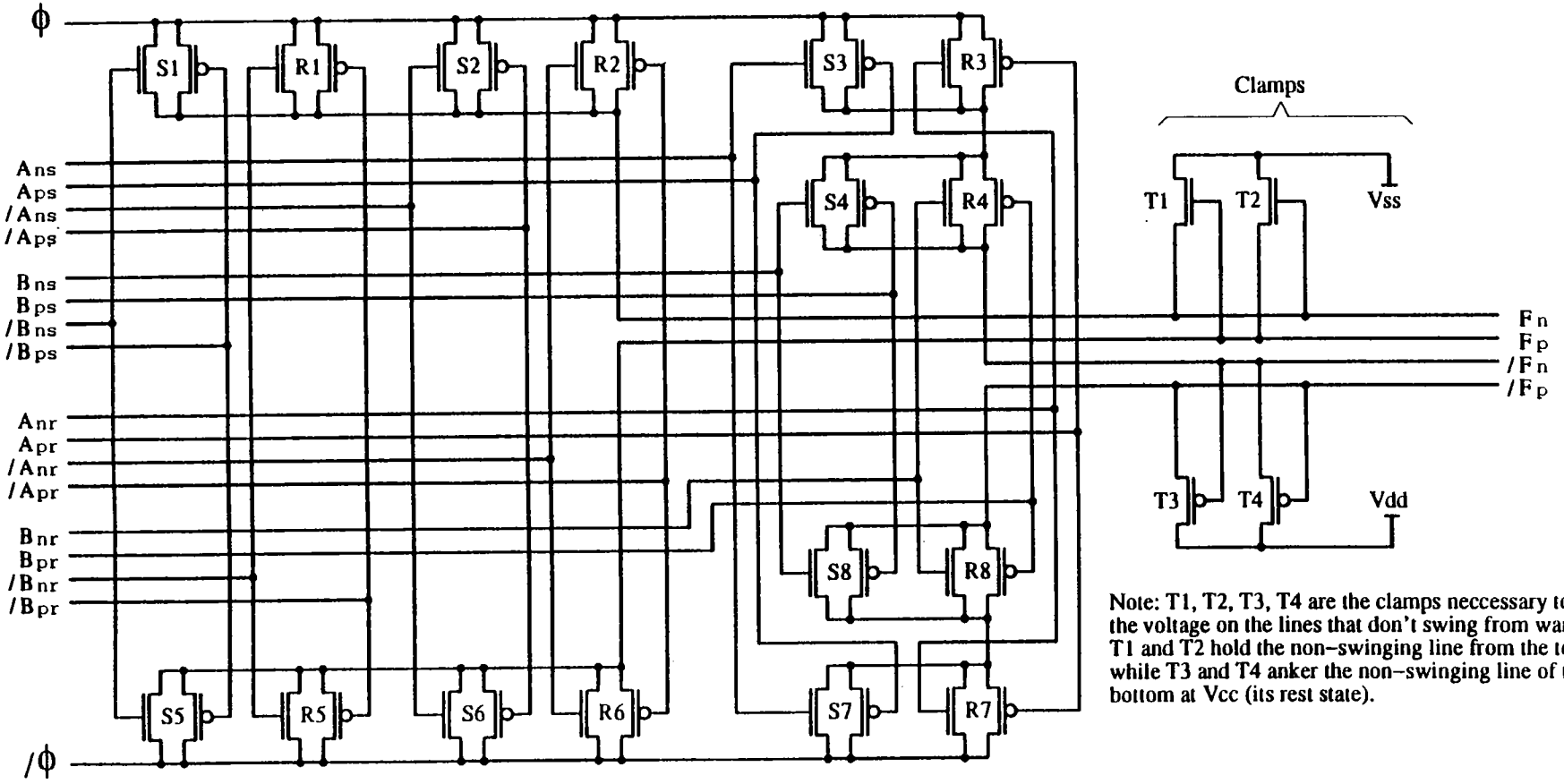
The register version below is SET by the slow rail going to 5V from 0V. The slow rail REST's at 0V. C1 and C2 are clamps that are neccessary to prevent the first stage internal nodes (1) and (2) from wandering. C3 and C4 are clamps neccessary to prevent the output nodes from wandering.



The register version below is SET by the slow rail going to 0V from 5V. The slow rail REST's at 5V. C1 and C2 are clamps that are neccessary to prevent the first stage internal nodes (1) and (2) from wandering. No output clamps are needed in this version because when the Slow Rail goes to RESET level of 5V, internal nodes (1) and (2) will go to 4V and that is enough to turn ON the N-Channel devices of P1 and P2 and hence these devices will anker the output nodes to 0V thus providing the clamping action and preventing output node wandering.



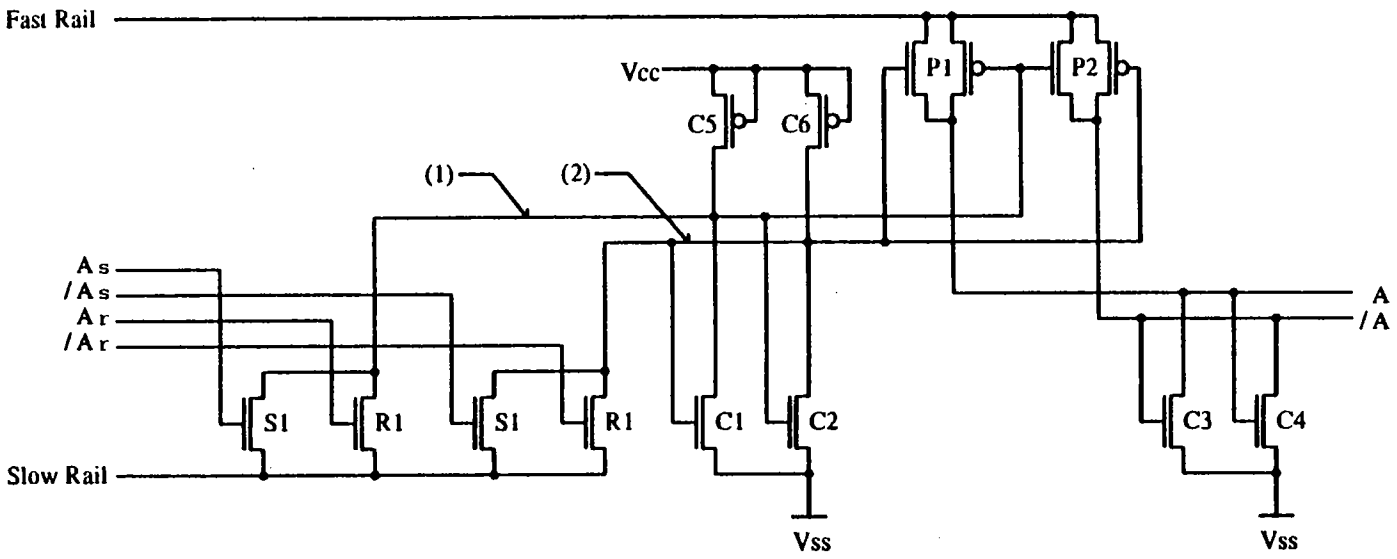
Below is figure 1 of the October 9th paper with the Clamps added

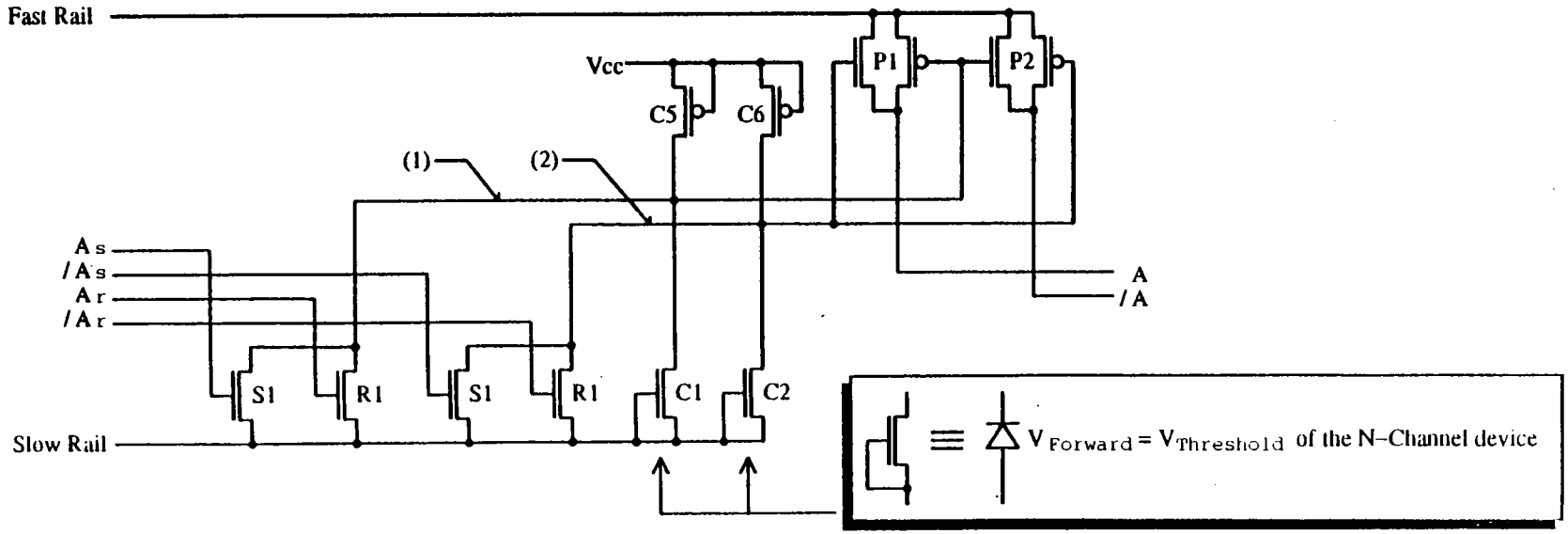


Note: T1, T2, T3, T4 are the clamps neccessary to keep the voltage on the lines that don't swing from wandering. T1 and T2 hold the non-swinging line from the top at Vss while T3 and T4 anker the non-swinging line of the bottom at Vcc (its rest state).

In this page there is the addition of how to prevent Latch–up from occuring in the new circuits.

For both versions of the new circuit, proper operation depends on boot strapping the voltage of the node (1) or (2) that is at 4V to a higher voltage. If the boot stapping increases the voltage significantly, Latch–up might occur in the circuit. To prevent this from happining, we add two more clamps, C5 and C6,to these nodes. sC5 and C6 make sure that the voltage at (1) and (2) never exceeds Vcc by more than one Threshold voltage (~1V). Through proper device sizing we can can control the boot strapping action so that the voltage will not rise to dangerous levels and hence the dissipative action of C5 and C6 will not be needed. They are included however as a sort of insurance and therefore the dissipation that they will contribute in practice is negligiable.





11 Rail Driver circuit Explanation

The following will explain the operation of the circuit that is used to swing a rail. As mentioned earlier, the rail would look like a capacitor from the prospective of this driver. The period of the reference clock determines the period of the rail oscillations. The rise and fall times of the rail are determined by the effective capacitance of the rail and by the inductor L1.

We start with the driven rail initially at V_{SS}. When the reference clock goes high, the output of the inverter I1 goes high. This triggers R1 and Q goes high. Going through the OR gate, OR1, a high reaches the gate driver G1 and the gate driver generates an output that is enough to turn the Power N-Channel MOSFET's, N1 and N2 on. This effectively connects the Rail to the inductor L1 and the rail voltage starts swinging towards V_{DD} and the inductor current would start to build up to charge the rail. When the rail reaches its peak voltage, close to V_{DD}, the current in the inductor reaches 0 and then starts to increase in the reverse direction. Since even the best MOSFET has some resistance when it is on, the current building up in the reverse direction will give rise to a potential across both N1 and N2. This will cause the output of the comparator C1 to go high thus resetting the register R1 which will turn off N1 and N2. So as soon as the voltage of the rail reaches the peak, the circuit will switch OFF N1 and N2 to stop the swinging. Due to unavoidable dissipation, the rail will not make it all the way to V_{DD}. However, when R1 resets, it sets R2 that turns on the current source H1. The current source will dump charge into the rail until the rail reaches V_{DD}. When the reference clock goes high, R2 is reset so that H1 is disabled.

We use current sources instead of CMOS devices to restore the rail peak voltage to its rest level for one important reason. If we use switches, then the rise time will be determined by the RC time constant of the chip and most of the dissipation will happen in the transistors that do the computation inside the chip. Using current sources, the dissipation will happen in the current sources that are usually off-chip, not to mention that the dissipation will be spread out over a longer time thus reducing component stresses.

In our circuit, we use two N-channel power MOSFET's connected in series to connect and disconnect the rail from the inductor at the right time. The reason for using two in series is a practical one. Manufacturers of Power MOSFET's connect the body to one of the terminals thus creating a parasitic diode that is in parallel with the device. Therefore, a discrete power MOSFET when off can prevent the current from flowing only in one direction. In the other direction, the parasitic diode will conduct even when the device is off. Putting 2 MOSFET's in series and pointing them in opposite direction, as shown, allows us to make a switch that will prevent current conduction in either direction when off and will conduct current in both directions when on.

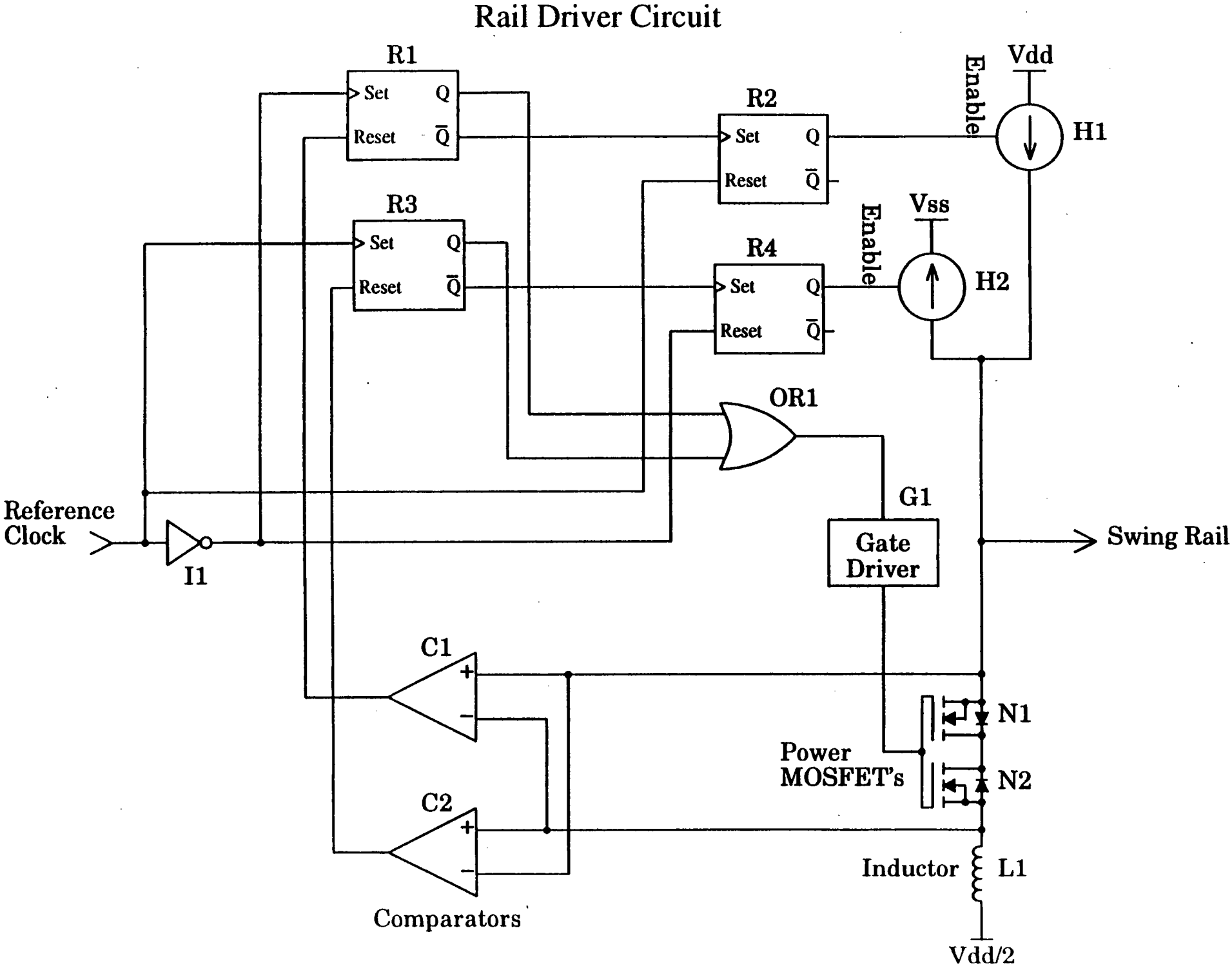
To turn the power MOSFET's on and off, we need to charge and discharge

their gate capacitors. Since power MOSFET's have a relatively large gate capacitance, this charging and discharging could easily become the dominant dissipation mechanism in a CRL system if done conventionally. We solve this by driving the the gates of the Power MOSFET's with a CRL rail driver (that is, replace the Gate Driver box, G1, with a Rail Driver Circuit). As we saw, the function of the Rail Driver Circuit is to generate a low-impedance, controlled swing at its output in response to a square wave at the reference clock input with minimum dissipation. If we replace the Gate Driver module, G1, in the Rail Driver Circuit with a copy of the full Rail Driver Circuit then we would have achieved our goal of providing low impedance drive to the Power MOSFET's gates with minimum power dissipation. We could recursively do the above until the energy saved by one more sub level is offset by the minimum energy that is necessary for operation of the Rail Driver control circuitry. At this point, the inner most level would have a conventional CMOS buffer inside the Gate Driver G1. The dissipation of this conventional buffer would be negligible because of significant current down scaling with every recursive level.

It is important to mention that the rail driver circuit could reside on-chip. The only component that must reside off-chip is the inductor L1. By integrating the Rail Driver Circuit on-chip, we greatly simplify the design of CRL logic. Engineers would assemble CRL IC's just as they do currently with conventional off-shelf CMOS. They simple have to provide $V_{DD}/2$ DC supply and an inductor for every rail in their CRL system without having to worry about the details of CRL rail control as it would be hidden inside the CRL chip. In addition it is possible to build power MOSFET's on-chip that do not have the parasitic diode in parallel with the device and therefore we would need only one Power MOSFET per rail. Also, the efficiency of the proposed rail driver depends highly on the propagation delays of the components. Slow components would shut down the MOSFET a significant time after the swing passes through the peak voltage and therefore would let some charge return to the rail leading to dissipation. On-chip components would have the best response time to minimize this kind of dissipation.

Conventional power supplies maintain a predetermined voltage at their outputs within a wide range of output currents. They supply a large amount of current if the output voltage drops slightly and they sink a large amount of the current if their output voltage where to be elevated. For maximum power saving, the DC power supply used for $V_{DD}/2$ has to be a little different. In conventional system, average as well as instantaneous power always flows from the power supply to the circuit. In a CRL system, the average power still flows from the power supply to the system to compensate for unavoidable dissipation. Instantaneous power however, moves back and forth between the power supply and the system. Power flowing into the supply should not be dissipated to maintain the predetermined output voltage. Rather, the energy must be stored even at the cost of allowing the output voltage to increase slightly. To achieve this, a CRL DC power supply would have a very large

capacitor at its output just as conventional power supplies. It would differ from conventional supplies in that its control electronics would supply charge to that capacitor to keep the *average, and not the instantaneous,* output voltage constant. It is significantly easier, and cheaper, to build switching power supplies that maintain constant average output voltage rather than the stricter demand of maintaining constant instantaneous output voltage.


Rail Driver Circuit
R1
Set
Reset
Q
$\bar{Q}$
R2
R3
R4
Vdd
Vss
Enable
H1
H2
OR1
G1
Gate Driver
Reference Clock
I1
Swing Rail
C1
C2
Comparators
Power MOSFET's
N1
N2
Inductor
L1
Vdd/2

12 CRL Carry-Save Adder for use in Fast Multipliers

The following is an explanation of how to build a CRL carry-save adder. I will refer to the accompanying diagram during the explanation. The example in the diagram is for a 3-bit adder with a single cycle throughput. The blocks used in the diagram are all CRL function blocks just like the ones in figure 3 of the october 9th paper. The arrows indicate if the block is part of the forward or reverse pipeline. The set inputs of the blocks are on the right and the reset are on the left. There are six different kinds of blocks used in the adder. The first is M1 and it adds bits A and B and outputs the sum bit (S) and a carry (Cout). The second is M2 and M3 and add A, B, and a carry-in (c) and output the sum bit (S) and a carry bit (Cout). The third is N1. It subtracts A from S and produces the result B and a borrow bit (Cout). The forth is N2 and it subtracts A from S taking in a borrow-in (C) and produces the result S and a borrow out (Cout). The fifth is N3 and it is identical to N2 but without the borrow-out (Cout). All the remaining blocks are of the sixth type and they are register blocks. They take a bit on a D input and produce a copy of it on the Q output. Data is fed from the top of the diagram and proceeds in a pipelined fashion to the bottom.

For CRL to work all operations have to be reversible. At a higher level the 3-bit addition should be reversible and therefore at least on of the input operands (we choose a0,...,a3) have to be passed on to the output in addition to the sum bits (S0,...,S3) in order to reverse the addition later on. In addition, each operation internal to the 3-bit addition, such as a 1-bit addition, have to be reversed as well.

We start by adding a0 and b0 in M1. We now have the first bit of the result s0 and we store it in a register pipeline so that it will reach the output at the same time as the other result bits. The carry bit and a1 and b1 get added in M2. Unfortunately we are unable to produce the carry for the reset side at the moment (not enough information available) so we start a register pipeline that stores the (Cout) from M1 and we use the reverse direction of this register pipeline to generate C for resetting M2. In essence we have moved the problem to providing Dr for the last reverse direction register in the pipeline. It is this idea of moving the problem (or postponing it) by using CRL register pipelines that should be mentioned.

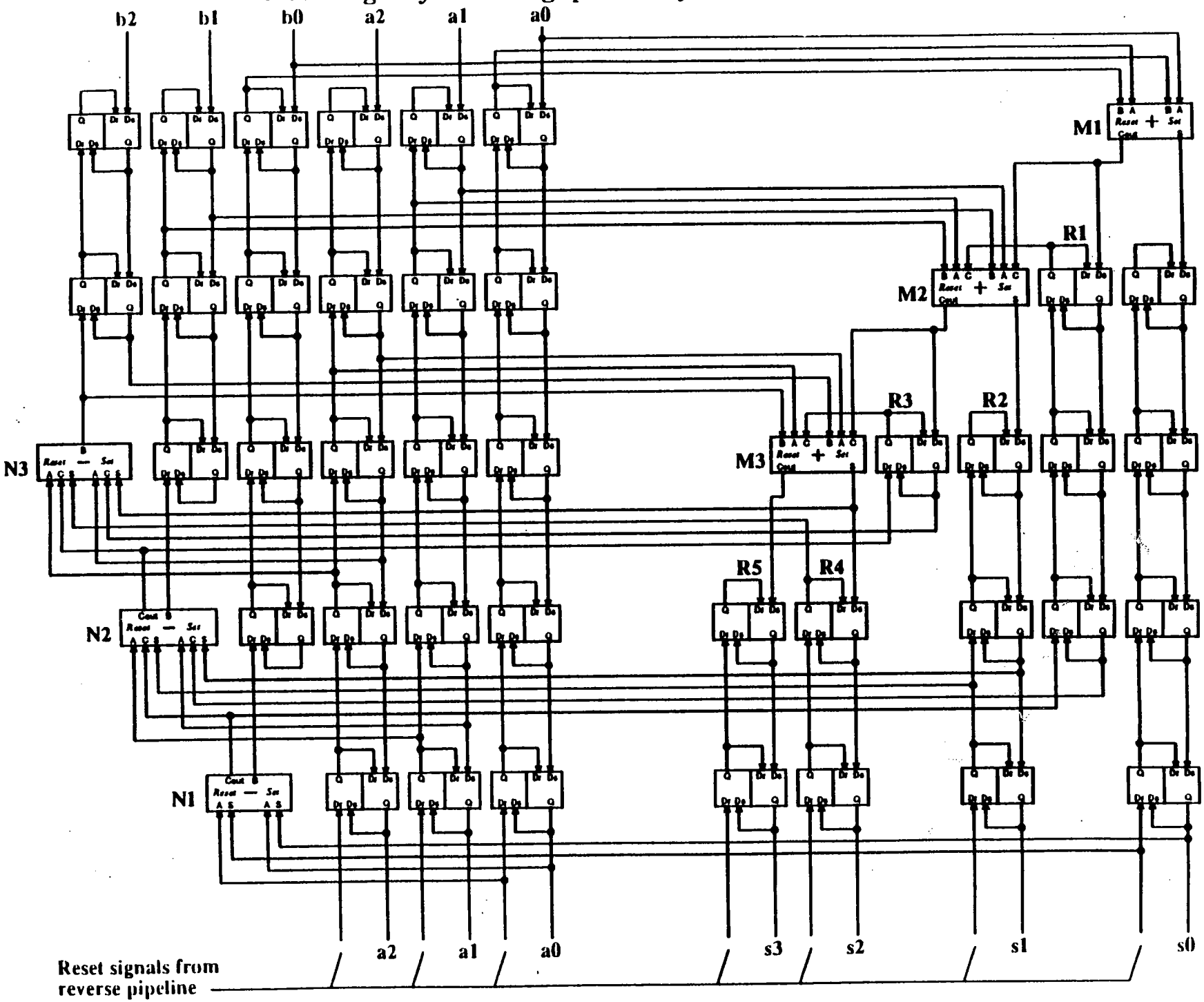
The same thing happens with M3 needing to get C for resetting and we store Cout of M2 in a CRL register so that we can provide it for the resetting side of M3 when needed. A this point however, the full 3-bit addition has been completed (s0,...,s3) are generated already and we could now start the reverse add (subtract). The first operation to reverse is that performed in M3 and it is reversed in N3. N3 takes the sum from M3 (s2) and subtracts from it a2 and the Cout of M2 that was stored is the register R1. This N3 produces a copy of b2 in the reverse direction that could be used to satisfy the reset input of the pipeline that stored b2 and therefore we can

stop that pipeline there as we don't need it to go on any further. In N2 we reverse the action of M2 and produce b1 and Cout from M2 and therefore we have produced the pieces of information that were needed to stop the b1 pipeline and the carry pipeline starting with R1. Finally, N1 is successful in producing b0 and Cout of M1 thus killing the last carry pipeline as well as stopping b0 pipeline and we end up with the sum (s0,...,s2) and (a0,...,a2) which are the minimum information set to reverse that addition!

Note that this adder can accept a new pair of input data every cycle and output the sum and a copy of one operand $2N-1$ cycles later at the output for an N-bit addition. Because one can start the addition of two new numbers every cycle without having to wait for the carry to propagate all the way to the end, this is called a carry-save added. Carry save adders is a well known method to add a bunch of numbers, our contribution is making one in CRL with the same throughput. Carry-save adder are an essential ingredient of fast parallel multipliers. A full schematic of a 4x4 multiplier using the above adder technique have been designed but is rather large to be describe easily.

The importance of the above adder is the technique of moving the problem of reversing a function to a latter time until you have enough information to do so. Knowing the above, it should be possible for people to reproduce the multiplier.

CRL Single cycle throughput Carry–Save 3–Bit Adder



File:verilog/radd2.ps