Asymptotically Zero Energy Split-Level Charge Recovery Logic

Saed G. Younis younis@ai.mit.edu
Thomas F. Knight, Jr. tk@ai.mit.edu

M.I.T. Artificial Intelligence Laboratory
545 Technology Square, Cambridge, MA 02139

Abstract [1]

As the clock and logic speeds increase, the energy requirements of CMOS circuits are rapidly becoming a major concern in the design of personal information systems and large computers. Earlier, we presented a new form of CMOS 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 [1]. The technique relied 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 functional inverse. Information necessary to uncompute the inverse is available from the subsequent inverse logic stage.

In this paper, we present a new and greatly simplified form of CRL, we call it Split-Level CRL (SCRL). Using split-level voltages, SCRL differs from our original CRL in that it needs only 2, instead of 16, times as many devices as conventional CMOS, requires only one wire for every signal, instead of 4, and actively drives all outputs during sampling. Furthermore, we will show how to construct SCRL circuits using only 2 external inductors for every chip.

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

Introduction

Power dissipation in conventional CMOS primarily occurs during device switching. One component of this dissipation is due to charging and discharging the gate capacitances through conducting, but slightly resistive, devices. 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, much longer than RC. An obvious conclusion is that energy 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 insist that two conditions apply throughout the operation of our circuit. Firstly, we forbid any device in our circuit from turning ON while a potential difference exists across it. Secondly, once the device is switched ON, the energy transfer through the device occurs in a controlled and gradual manner to prevent a potential from developing across it. These conditions place some interesting restrictions on the way we usually perform computations. 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 our low energy charge recovery logic. By low energy, or non-dissipative, we mean that the energy per computational step can be made arbitrarily small by spreading the computation over a longer period.

The paper starts by examining the topology and functionality of a simple SCRL gate, an inverter. We will then show how a general SCRL gate fits into a computational pipeline. We will proceed to show how to generate the swings on the rails using 2 external inductors. Finally, we will discuss how SCRL relaxes some of the device sizing constraints that apply to conventional CMOS circuits as well as illustrate some techniques through which the cost of irreversibility, in energy and complexity, could be minimized.

Split-Level CRL Gate

In this section we describe the topology and operation of an SCRL inverter. Extension to other gates is straight forward. A device-level diagram of the SCRL inverter is shown in Figure 1. It is identical to a conventional inverter except for the addition of a pass gate at the output and that the top and bottom rails are now driven by clocks rather than

[1] This research is supported in part by the Advanced Research Projects Agency under contracts N00014-91-J-1698 and DABT63-92-C-0039.

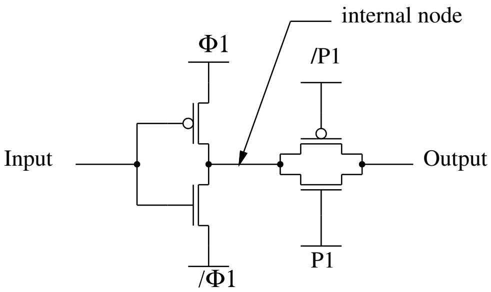

Figure 1: SCRL inverter.

V_{dd} and GND. We call the clock controlling the top rail ϕ_1 and that controlling the bottom rail $/\phi_1$. We refer to the clocks that control the pass gate as P_1 and $/P_1$.

Initially, the input, ϕ_1, $/\phi_1$, the output, and all internal nodes are at $V_{dd}/2$. In addition, P_1 is at GND and $/P_1$ is at V_{dd}, *i.e.,*the pass gate is turned off. After accepting a valid input, V_{dd} or GND, we turn the pass gate on by gradually swinging P_1 and $/P_1$ to V_{dd} and GND respectively. We now gradually swing ϕ_1 to V_{dd} and $/\phi_1$ to GND. The fact that both ϕ_1 and $/\phi_1$ start at $V_{dd}/2$ and *split* towards V_{dd} and GND respectively is the reason we call this family Split-Level Charge Recovery Logic (SCRL). If the input to the gate was at V_{dd} then the output would follow $/\phi_1$ to GND. If the input was at GND then the output would follow ϕ_1 to V_{dd}. We note that at the end of the ϕ_1, $/\phi_1$ swings, the output is the NOT of the input. The output is also actively driven and could now be sampled by another gate later in the pipeline.

After the output is sampled by a later gate, the pass gate of this inverter is turned off thus tri-stating the output. Following that, we return ϕ_1 and $/\phi_1$ to $V_{dd}/2$. This in effect restores all the nodes except the output to $V_{dd}/2$. We are now ready to accept a new input. Please note that allowing the input to change prior to resetting all the nodes to $V_{dd}/2$ could turn some devices on while there is a potential difference across them leading to dissipation.

Remember that the output is still at a valid logic level, not $V_{dd}/2$, and before turning on the pass gate we must restore the level of this output to $V_{dd}/2$ to prevent dissipative charge sharing. The promise is that at the point that the pass gate disconnected the output from the inverter, the output was connected to a different gate that has the job of restoring its level to $V_{dd}/2$. We will show how this is done in the following section.

Reversible Pipeline

The reason for not letting an SCRL restore its own output to $V_{dd}/2$ is to allow pipelining. Note that to non-dissipatively restore the output to $V_{dd}/2$, the input to the gate must be held constant during the splitting and restoration of its rails. The same restriction dictates that this gate does not restore itself before the subsequent gate in the pipeline restores itself and so on. This means that a new input to a pipeline must be held constant until the effect of this input propagates all the way to the end of the pipeline and until the restoration of the pipeline starting from the last stage reaches back to the first gate. This form of "pipelining" is obviously not very useful.

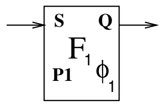

Figure 2: Abstraction box.

In this section we show how to connect SCRL gates, or stages, in a non-dissipative pipeline. The main purpose of this method of interconnection is to provide a way of restoring the level of gate outputs to $V_{dd}/2$ with the right timing. We build the pipeline out of copies of an abstraction box shown in Figure 2.

We think of this box as containing a parallel set of SCRL 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 SCRL gates internal to the box. The input to the box represents a bundle containing all the inputs of the gates internal to the box. The function computed by the box is identified by the letter in the center of the box. Finally, indicated at the bottom of the box are the clocks used to control both the split-level rails and the pass gate controls of all the SCRL gates internal to that box. A clock of ϕ_1 in the lower right corner indicates that the top rail is connected to ϕ_1 and the bottom rail to $/\phi_1$, while a clock of P_1 in the lower left corner indicates that the pass gate is on when P_1 is high.

Using this abstraction, Figure 3 illustrates how SCRL gates are connected to produce a non-dissipative pipeline. Note that the box with a function F^{-1} performs the inverse operation of the box with a function F. The computation proceeds from left to right in the top half of the pipeline and the "uncomputation" proceeds from right to left on the bottom half of the pipeline.

Each line linking SCRL gates is connected to the outputs of two different SCRL gates. For example, node (a) is connected to the output of F_1 and to the output of F_2^{-1}. There are two reasons why no logic fights occur between the gates driving the same line. The first is that when one gate is driving the line the other is tri-stated and visa-versa. The second is that during hand off, the voltages at the output of the gates is guaranteed to be equal. In this pipeline, the forward gates are responsible for gradually swinging an output line from $V_{dd}/2$ to V_{dd} or GND depending on the computation. The reverse pipeline is responsible for restoring the output line from the active levels to $V_{dd}/2$.

To avoid dissipation, the backward gates have to determine the value of the output that they are about to restore to $V_{dd}/2$ and set their internal node to that level before their pass gate is switched on, *i.e.,*before the line in handed off from the forward gate. To see how this works, we go through the events that occur after a new input, say a_0 is

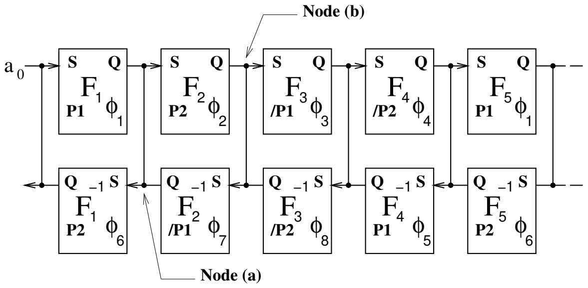

Figure 3: Non-dissipative multi-stage pipeline connection.

presented to the pipeline. First P_1 turns on the pass gate of F_1 and turns off the pass gate of F_2^{-1}. Next ϕ_1 splits setting node (a) to the valve $F_1(a_0)$. F_2 goes through similar transitions and produces $F_2(F_1(a_0))$ at node (b). Similarly F_2^{-1} produces $F_2^{-1}(F_2(F_1(a_0))) = F_1(a_0)$. Note that at this point the voltage levels at the outputs of F_1 and F_2^{-1} are at the same level which means that it is now safe to hand off node (a) to F_2^{-1} from F_1 by swinging P_1 low. After the hand off, we can restore F_1 by restoring ϕ_1. This could occur even without having to wait until F_2 is restored because F_2^{-1} is still holding node (a) at its valid value. After F_2 is restored, F_2^{-1} gradually restores node (a) to $V_{dd}/2$ and hands it over to F_1. The timing diagram for a four phase clocking scheme is shown in Figure 4. For $\phi_1 \ldots \phi_8$ in the figure, a high indicates when they are split and a low when they are restored. For $P_1 \ldots P_2$, a high is V_{dd} and a low is GND. With this pipeline, we are able to accept a new input every ϕ_1 without needing to wait for the restoration of later stages.

There remains one problem however. At the end of the pipeline, the input to F_5^{-1} is not restored and hence driving this line is dissipative. Furthermore, it could not be generated, as this is the place where reversibility is broken. This implies the fundamental limit that links information entropy with thermodynamic entropy. If at any moment a piece of information that is vital to reconstruct the past is lost, energy is dissipated. Fortunately, this dissipation occurring only at the end of a long pipeline and is negligible.

Rail Driver Circuit

As shown before, for a 4-phase SCRL implementation, we need a total of 20 clocks, or swinging rails. The previous implementation required a single inductor for every swinging rail and used the circuit shown in Figure 5.

A rail could be approximated by a capacitor in series with a resistor. The capacitor is the sum of the capacitances that the rail is driving and the resistor is the equivalent resistance of the devices through which the capacitances are driven. Suppose that the initial voltage on the rail was V_{init} and we want to swing the rail to V_{fin}. To start the swing, we connect the rail through an inductor

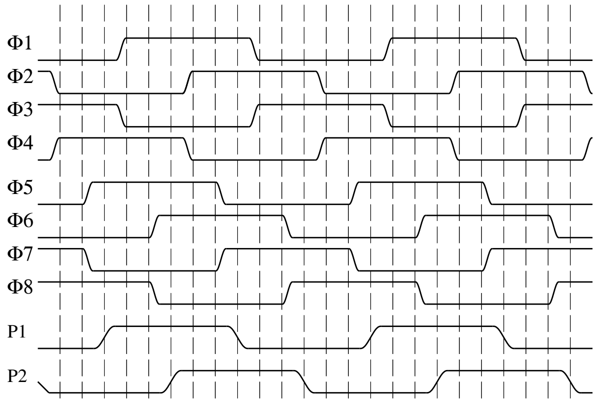

Figure 4: Rail timing for 4 phase SCRL.

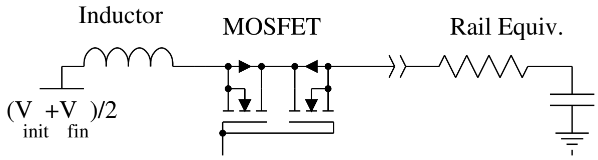

Figure 5: CRL rail driver circuit.

to a DC power supply at $(V_{init} + V_{fin})/2$. Current starts to build up in the inductor and the rail starts the swing towards V_{fin}. At the moment that the current drops back again to zero we disconnect the inductor. The rail should now be at V_{fin}. The action of connecting and disconnecting the rail is performed by the power MOSFET. Please note that the inductor is only necessary during the transition and is otherwise disconnected from the rail. Note further that the current in a disconnected inductor is zero. With this in mind, we should be able to multiplex the inductor among multiple rail circuits so long as these multiplexed rails do not have simultaneous transitions. Examining the timing diagram of Figure 4, we see that no more than two transitions occur simultaneously at any moment. From this we see that the maximum number of required external inductors is 2. Integrating everything but the inductors inside the silicon chip results in a chip with SCRL behavior with very little external components.

There are two difficulties in fabricating circuits with high energy recovery factors. The first is the Q of available inductors. While our derivation assumed an infinite inductor Q, realistic high frequency inductors at room temperature have Q's limited to the range of 100-200, placing an upper limit on the achieved energy recovery.

The second and more serious difficulty is the dissipation associated with controlling the action of the power MOSFET. Koller and Athas [2] have studied this problem and have shown that dissipation in the power switch forces the energy dissipation of the overall system to follow $f^{1/2}$ instead of the theoretical minimum that drops linearly with f. Fortunately, we can partially recover the linear rela-

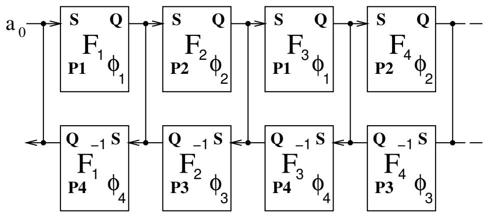

Figure 6: Two-Phase SCRL pipeline.

tionship between energy dissipation and operating frequency by employing a number of techniques described in [3].

SCRL Clocking Variants

In what follows, we will describe a number of alternatives for clocking SCRL circuits. These circuits differ primarily by the number of required phases and/or rails that are needed to control their operation. The pipeline described previously in this paper required four-phase clocking. This used four different clock phases in the forward pipeline and four others for the reverse pipeline. By four-phase we mean that the shortest feedback path in the pipeline has to span a minimum of four pipeline stages. Here, we will show how to construct SCRL circuits using two-phase clocking. Generalizing to three-phase, five-phase and six-phase clocking is a little tricky but not overly difficult and will not be included here for lack of space. One might simplistically think that less phases lead to less required rails. This unfortunately is not true because for some implementations the required phases are not symmetric and therefore the complement of a phase cannot be used to drive an oppositely timed rail. For this reason, the primary reason for reducing the number of phases is to minimize the number of stages for the shortest feedback path and to increase the throughput of the pipeline. Additionally, the lower the number of phases that an SCRL circuit uses, the easier it is to understand and apply.

Two-Phase SCRL

For the implementation that we will describe in the this sections the basic gate is the same as the one described in the "SCRL Gate" section of this paper. Figure 6 shows a pipeline of a two-phase SCRL implementation. The timing relationships among the rails are shown in Figure 7. Two-phase SCRL forfeits the benefit of always actively driving the nodes whenever they are sampled in exchange for achieving two-phase pipelining.

For $\phi_1 \ldots \phi_4$ in the timing diagram, a high indicates the time when ϕ and $/\phi$ are split and a low indicates when they are at $V_{dd}/2$. For $P_1 \ldots P_4$, a high indicates that P is at V_{dd} and $/P$ at GND while a low indicates that P is at GND and $/P$ at V_{dd}. The bottom two timing lines indicate the states of outputs driven by ϕ_1 and ϕ_2 gates. A high indicates when the output is at an valid level of V_{dd} or GND, while a low indicates that the output is at $V_{dd}/2$. The shaded regions in the timing diagrams indicate the times at which the signals are not being actively driven, *i.e.*,floating at an active level.

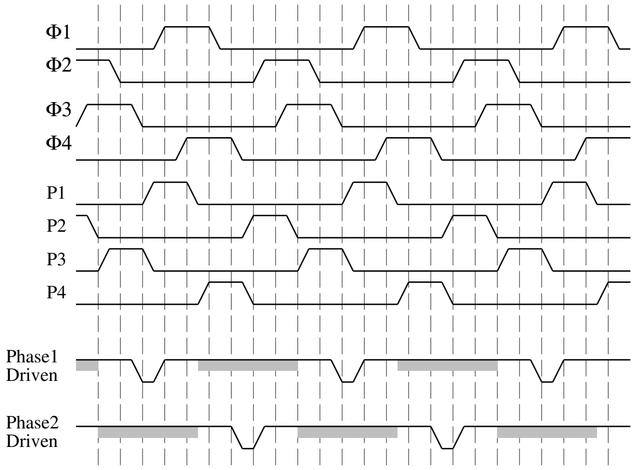

Figure 7: Rail timing for 2 phase SCRL.

We note here that clocking schemes with more than two phases always have actively driven nodes.

Non-Inverting Stage

Since the basic SCRL gate mimics that of conventional CMOS we find out that it is not possible to pass a signal through an SCRL stage without inverting it. For some circuits it is necessary to receive both the true and complement of a logical signal simultaneously at the inputs of a logic gate. Starting with a single signal, it is not possible to have its true and complement arrive at a later stage simultaneously, given the circuits we have described so far. In order to pass a signal without inversion we substitute the basic SCRL gate with the one shown in Figure 8. Please note that this buffer requires an additional set of controlling clocks we call "Fast ϕ_1" and "Fast $/\phi_1$" for a ϕ_1 gate. The restriction on Fast ϕ_1 is that is splits *immediately after* ϕ_1 splits and that is restores *just before* ϕ_1 restores. In other words, the transitions of ϕ_1 contain within them the transitions of Fast ϕ_1. For stages where we want to pass a signal without inversion, we use a gate similar to the one in Figure 8 and we clock its fast clocks according to the relations described.

In place of the inverters in Figure 1 and Figure 8 one can put any CMOS gate such as NAND, NOR etc. We can see that an additional benefit of a non-inverting SCRL gate, is that it allows each functional block to have a 2-level logic implementation. This generally aids in reducing the storage buffers that are sometimes needed for reversibility.

Another benefit of having a 2-level SCRL has to do with optimal step-up ratio of logic gates. It is well known that a CMOS inverter made out of the minimum size devices can optimally drive between 3-5 other inverters of the same size. A minimum sized inverter driving more than this optimal step-up number of loads similar to its size would have a larger delay. Since the energy saving of SCRL is referenced to the maximum operating frequency of a similar circuit in conventional CMOS, this longer delay leads to less energy savings. For an inverter

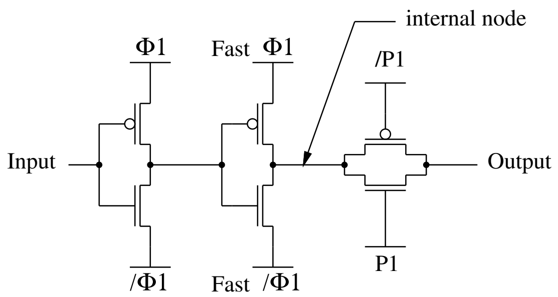

Figure 8: Non-Inverting SCRL Gate.

to drive more than 3-5 loads and maintain the same speed, it must be made out of larger sized devices. Unfortunately, larger devices have larger input gate capacitances and hence present a larger load to the gates that are driving them. To see how this could be a problem, let us consider building a multiplier out of an array of identical 1-Bit SCRL adder gates. The multiplier would consists of an array of gates in which each gate takes its inputs from a previous *identical* gate and provides on its output the data for the inputs of another *identical* gate. Typically in these arrangements, each output would fan-out to drive more than 3-5 loads because each input to a gate feeds a number of devices internal to that gate. For SCRL, just as for CMOS, having an output drive more than 3-5 loads its size is not optimal. As mentioned earlier, increasing the driving capability of a gate so as to be able to drive the loads, i.e. by doubling the width of the devices used in it, also increases the input capacitance, and hence the load, that this gate presents to the *identical gate* driving it. By attempting to increase the driving capability, we also increased the loads, and thus lost the benefit that we where attempting to gain.

Having 2-level SCRL allows for increasing the driving capability of a gate without increasing the load it presents to the other gates. This is done by performing most of the computations in the first level and then using the second level to provide the buffering. For this reason the first stage can consist primarily of minimum sized devices, and thus present the minimum load for the previous gate, while the second stage is made of devices 3-5 times the minimum size to give optimal driving capability.

Finally, the timing diagram of a two-phase SCRL with Fast clocks is shown in Figure 9. The figure indicates the position of the transitions of the fast rails using the dashed lines.

Device Sizing in SCRL Gates

In conventional CMOS, the devices within a gate are sized to accommodate the worst case delay in the system. This is because the system clock is directly related to the worst case propagation delay in the system. If the propagation delay of a CMOS gate located between 2 registers is longer than the clock cycle, the circuit will not work. Note however that in SCRL the clock period is always considerably longer than the worst case propagation delay

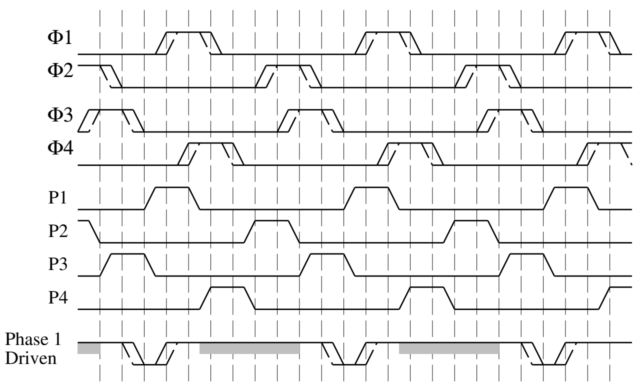

Figure 9: Timing diagram of Two-phase SCRL with fast rails for non-inverting stages.

and therefore the circuit will always function correctly. The reason why we are concerned with optimal sizing for SCRL gates is because the energy savings is directly related to the ratio of the maximum possible operating frequency of the circuit to the intentionally much lower frequency at which we will operate the circuit. Therefore, if we wanted to operate at 1 MHz, then building circuits that can potentially run at 100 MHz would save us 10 times more energy than if we did a poor job and build a circuit that could operate up to 10 MHz only.

There is however one subtle, but *very important* difference. Since it is the average, not the instantaneous, power than we are concerned with, sizing of devices in SCRL gates should be optimized for the *average, not the worst,* case. For example, it is considered poor design practice to use carry-ripple adders in conventional CMOS circuits. An 8-bit carry-ripple adder consists of 8 1-bit adders. There, it is possible that the carry out of the 1-bit adder in the least significant position affect the result out of the 1-bit adder in the most significant position. To guarantee correct operation in conventional CMOS, the designer has to size his devices, or slow down the system clock, to accommodate the case in which the carry propagates from the least significant bit to the most significant bit, *i.e.,*the worst case. For an 8-bit adder this could correspond to 8 gate delays. Statistically however, a carry in an adder propagates 0.6 bit positions on the average and only makes it all the way across about 1/1600th of the time in an 8-bit carry-ripple adder. Therefore a properly sized SCRL would be optimized for 0.6 carry propagate and would only consume more energy whenever the very infrequent worst case occurs. The importance of this cannot be over emphasized. The same could be applied to sizing paralleled NMOS transistors in a NOR gate. Unlike conventional CMOS where the sizes of paralleled devices are not decreased because of the worst case of only one device being ON, paralleled devices in SCRL could be made smaller because *on the average* more than one of the paralleled devices would be on at any given time. This would improve the energy savings factor as well as require less device area to implement.

Lowering Irreversibility Cost

In CRL and SCRL the reverse pipeline is required to accurately provide a delayed copy of the inputs that were used in the forward pipeline. Without the reverse pipeline we do not have enough information to always correctly compute the delayed copy of the inputs that are required to non-dissipatively reset the stages in the forward pipeline. The penalty of erroneously computing a delayed copy of these inputs is to dissipate energy similar to conventional CMOS for every erroneous bit. Unfortunately, there are situations in which providing the inverse of a function in the forward pipeline is not feasible or at least cumbersome. Luckily all is not lost since in most of these cases we could apply a number of techniques that would make the dissipation associated with this irreversibility minimal.

The first relies on the observation that we are mainly concerned in reducing the *average, not the instantaneous*, power consumption. Gates that computed the inverse function of the gates in the forward direction produced a correct copy of the inputs all the time. Without these inverse gates, we cannot guarantee to be correct all the time. In certain applications however, we can guarantee to be correct *most* of the time. Since dissipative events only occur whenever we guess wrong, being correct most of the time results in substantial energy savings when compared to conventional CMOS *without* the need for reversibility. To illustrate this we consider an example of an 8-Input NAND gate. This gate outputs a FALSE if and only if all the inputs where TRUE. Otherwise, this gate outputs a TRUE. Assuming that the input bits are random, the probability of the output of this gate being at TRUE is $255/256 = 0.996$. If we always assume the output to be at TRUE, then we will have a dissipative event, caused by a wrong prediction, only 0.3% of the time. In the pipeline in Figure 3 let the F_1 be this 8 input NAND gate. Then we can omit F_2^{-1}, assume that this omitted inverse gate output a FALSE all the time, and be right 99.6% of the time. This could be important in situations in which the computation of F_2^{-1} is not feasible or otherwise cumbersome.

The second technique concerns the way multi-stage buffering is done in CMOS and in SCRL. To drive a large load in CMOS, one must go through a number of progressively larger devices with each device driving another that is slightly larger than itself until the last one in the chain is large enough to drive the load. In SCRL, each larger stage would be paralleled by another stage of comparable size in the reverse direction. If reversibility was broken immediately after the largest stage then dissipation would be large owing to the fact that the input capacitance of the large reverse gate is significant. To alleviate the problem, we must proceed with the pipeline beyond the last stage with inverters in the forward and reverse direction scaling down the size at each successive stage until we reach the minimum size possible. If reversibility is broken immediately after this minimum size stage, dissipation would be minimized due to the much smaller input capacitances of the reverse stage. This is shown in Figure 10.

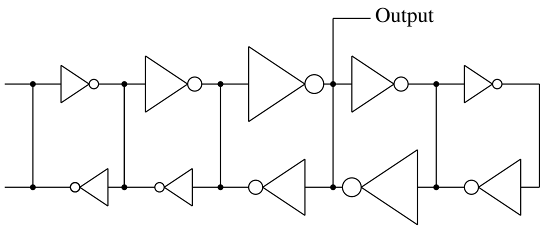

Figure 10: Scaling down stage sizes before breaking reversibility in SCRL.

Demonstration Chip

To verify the concepts in this paper, we have completed a design for an 8x8 bit multiplier chip using SCRL techniques outlined above. Among the concepts included in this chip is the use of carry-ripple adders in the forward direction and carry-ripple subtractors in the reverse direction, as well as the sizing of devices within the chip to optimize for the average, instead of the worst, case. The chip also uses predictor buffers where reversibility is broken, and employs the scale-up scale-down technique in sizing the pad driver chains. The chip is currently being fabricated by MOSIS.

Conclusion

We have demonstrated a new techniques for constructing charge recovery logic. The new form, SCRL, enjoys the same asymptotically near-zero dissipation property as its CRL predecessor. Key ideas include the reduction in the number of devices per gate and the number of wires per signal by using split-level clock rails and by distributing the tasks of splitting and restoration among different gates. We believe that the simplicity of SCRL gate design, nearly identical to conventional CMOS, makes its adoption more acceptable when compared to our previous implementation. A demonstration chip employing SCRL techniques is currently in fabrication.

References

[1] Younis, S., Knight, T., F., "Practical Implementation of Charge Recovering Asymptotically Zero Power CMOS", *Proceedings of the 1993 Symposium on Integrated Systems,* MIT Press, 1993, pp. 234-250.

[2] Athas, W. C., Koller, J. G., and Svensson, L. , "An Energy-Efficient CMOS Line Driver Using Adiabatic Switching," *Proc. 1994 IEEE Great Lakes Symposium on VLSI,* March, 1994.

[3] Younis, S., "Asymptotically Zero Energy Computing with Split-Level Charge Recovery Logic", PhD Thesis, June 1994 (in preparation).