Pipelined Architecture for Computational Nanotechnology

R. Mercer, M. Ebel
Martin Marietta Inc.
Information Systems Division
Orlando, FL 32859-0385
Tel: 407-356-5610
Fax: 407-356-8949

R. DeMara
Dept. of Electrical and Computer Engr.
University of Central Florida
Orlando, FL 32816-2450
Tel: 407-823-5916
Fax: 407-823-5835
rfd@engr.ucf.edu

Abstract

Nanomechanical computing elements which are scalable in terms of input size and depth of propagation path are analyzed using a bounded continuum model. Boolean logic functions of NOT, AND, OR, and XOR are realized using helical latch, reset spring, and translating rod assemblies. Building upon these components a design for two-level logic operations is presented. The helical latching mechanism calculates the Boolean output function as a positional displacement from a known reset state, which occurs exactly once during each instruction cycle. To balance forces a symmetrical rotor is used to counteract applied forces by replicating input rods. This has the beneficial side-effect of providing intrinsic fault-detection capability within a gate and also decreases the rotation required for a full cycle from 360 degrees to 180 degrees.

This design is further enhanced to allow operations of arbitrary word length by subdividing the logic disc into sectors where each sector contains all the components necessary to operate on a single bit. The benefits of increasing the disc diameter needed for additional bits include a further reduction in disc cycle rotation as a result of subdividing the disc into sectors. Since the inputs are sampled sequentially, throughput of resultants can be increased directly by pipelining multiple bit operands. For n inputs per logic gate, the maximum speedup for a single level of logic is $(n+2)$. Generally, speedup is bounded by $(n+2)/p$ where p denotes the number of cycles between initiations of the pipe.

Keywords: Nanocomputation, Incremental Boolean Logic Techniques, Radial Symmetry Nanomechanical Device, Logic Rotor, Helical Latch.

1 Introduction

Since the advent of the electronic computer, a primary performance goal has been to design processing and memory devices which are increasingly cost-effective: faster, smaller, more reliable, and inexpensive. In this paper, highly scalable nanomechanical processing elements with increased efficiency are described using molecular design principles. While previously proposed nanomechanical computing elements relied solely on translational displacements to calculate output values, an enhanced design is presented which takes advantage of radial symmetry and rotational motion. Benefits include the availability of gates which are readily scalable in terms of their number of inputs and in depth of logic stages from input to output. Essentially, the inputs to each gate are isolated from subsequent stages of logic by selectively driving the output indicators without the use of fixed linkages from preceding stages of logic.

1.1 Previous Work

Computing systems designed using molecular nanotechnology were proposed by Drexler [1]. The advantages of this type of computing system are derived from the nanometer-scale geomtries of the switching elements. Drexler's design employed logic gates which rely primarily on translational movement of rods to realize a blocking/non-blocking interlock mechanism. The mechanical motion which operated the interlock was provided solely by the drive force of the rods to each gate. Under this scheme, the output from one logic gate is directly connected to the input of another gate, so that computation through subsequent stages of logic relies on mechanical linkages from one stage to the next.

In this paper, we propose enhancements to avoid the potential drawbacks of mechanically-linked stages of logic. Basically, we have decoupled the movement of the input logic rods from the force that drives the computing mechanism. Each gate is partitioned into stages which perform separate input, logic, and output functions without requiring interconnected rods. We will show that a partitioned design can avert potential race conditions that a linked circuit would encounter.

1.2 Components for Nanomechanical Computation

The proposed logic gates have several mechanical components in common. For background and clarity these components will be described separately in this section. It is assumed the scale of the mechanisms presented are

similar to those offered by Drexler [1].

1.2.1 Compression Springs

Mechanical devices which perform repetitive operations require a means to obtain a predictable initial state at the start of each cycle. An effective method for producing this type of resetting and retaining capability is to use *compression springs*. The springs are assumed to exert a constant force throughout their range of motion and will be depicted schematically in diagrams as their macro-scale counterparts.

Springs are considered to be loaded in their initial position which is referred to as a *preload*, usually denoting a 0 logic state. When located within its housing, it will be compressed (preloaded) so that it always exerts a force onto the rod. When an additional drive force of sufficient strength is exerted on the preloaded spring, the preload is overcome and translation in the rod occurs. As the force is relaxed, the spring becomes the dominant force and the rod is reset. An equivalent implementation using magnets can be achieved by facing the same pole of two magnets towards each other; creating a repulsive force, pushing the rod away from the housing.

1.2.2 Actuating Rods

A variety of mechanical *rods*, such as input and output rods, are used as a medium to accept and pass binary values to and from the logic gate. They are assumed to be rigid and are constrained to translate linearly in their respective housings. The rods communicate the current state of the gate via a ramped surface. The geometries proposed for the ramps must be sufficiently large such that arbitrary shapes and angles of inclination are not affected by the bonding properties of the material.

Other rods and ramped surfaces are used in the operation of the logic gate. The *reset rod* is a stationary rod with a ramped surface which is used to reset the gate to a known state at the beginning of each cycle. *Momentum Reversing Ramps* (MRRs) are also rigid, stationary rods. They are attached to both stators and rotors within the gate and are used to ensure the moving rods are reset prior to accepting or acting on subsequent inputs.

1.3 Component Housings

A rigid stationary structure is used as the stator for both the input and output stages of the logic gate. They are used to guide the movement of the input and output rods, as well as supplying a rigid base for the reset rod and momentum reversing ramps.

The *rotor* is a rigid housing which is attached to an axially-connected *shaft*. The rotor spins with respect to the stationary *stator* element and houses the auxillary components necessary to perform logic functions on the inputs. The shaft of the rotor is assumed to be driven by a nanomechanical motor similar to those proposed by Drexler having rim speeds on the order of $1000m/sec$ [1].

2 Design of Logic Gate

A rotational logic gate is proposed which is comprised of 3 stages: an *input stage*, a *logic stage* and an *output stage*. These stages are shown from top to bottom, respectively, in Figure 1. The stages function independently of each other except for interaction by means of interconnecting rods and ramps.

2.1 Input Stage

The input stage accepts binary inputs from an external drive source which either translates the rods linearly towards or away from the logic stage. This action is depicted in Figure 1 and Figure 3. When translation towards the logic stage occurs, the resultant input value is considered to be a logic "1." Alternatively, if there is no translation of the input rod towards the logic stage, the input is read as logic "0." The default position of each input rod is a logic 0 state which is retained by a preload from a compression spring. Only when this preload is overcome by a larger drive force, the input rod may be translated to a 1 state. To ensure that the input rods are properly reset to a value of 0 prior to accepting subsequent inputs, a MRR located on the logic stage is integrated into the input rod. The contact between the ramped surface of the input rod and the components of the logic stage serves as a means of communicating or passing its state value on to the logic stage. The total number of input rods required is given by $n \times b$ where n is the number of operands and b is the number of bits in the word.

The housing contains a clearance hole which permits the rotor driveshaft to be driven by an external motor. The input rods are constrained by the housing to operate in a linear motion. The input stage also incorporates a *reset ramp* which protrudes downwards towards the logic stage. This ramp on the stationary housing is located so that it only contacts the reset pin when the logic rotor rotates directly beneath it as shown in Figure 1 and Figure 2. The sole purpose of the ramp is to reset the *helical latch* storage mechanism in the logic stage. Every time the logic stage rotates beneath the reset ramp, the helical latch is preset to a known condition. This preset value may assume either a 1 or 0 state depending upon the boolean logic function being implemented.

2.2 Logic Stage

As shown in Figure 1, the logic stage is located immediately below the input stage. The logic stage consists of a rotating disc called the *logic rotor* which is attached to the main driveshaft. Its primary purpose is to house the helical latch which performs boolean functions while sampling inputs received from the input stage.

The helical latch is a translating and rotating rod which can assume binary values as shown in Figure 3. If the ramp on the input rod makes contact with the tip of the helical latch then the resulting movement is a translation downward which causes a clockwise rotation in a corkscrew-like movement along the helically-shaped surface of the latch. As shown in Figure 3, this movement locks the latching pin into its next state with the

assistance of the MRR located on the output stage. The function to be implemented determines the geometry of the helical surface. In other words, an XOR gate will have a different helical surface contour than an AND gate. Regardless of the gate function, the helical latch is retained in its current state by a compression spring.

As shown in Figure 1, a stationary pin is located along the identical radius and oriented in the same direction as the helical latch. It is engaged with the input rod immediately following the sampling of that input by the helical latch. It ensures that the input rod is returned to a value of 0 prior to the next cycle by the action of the reset pin. The pin height is selected such that the tip only contacts those input rods whose values are 1. The reset pin assists the compression spring to reset the input rod prior to the next cycle. The distance at which the reset rod lags the helical latch is specifically selected so that no reset pin contact occurs in coincidence with the externally applied drive force on the input rods.

The logic rotor may house multiple helical latches; typically there will be a helical latch for each bit in a word. Likewise the logic rotor houses one reset pin for each bit in a word; one corresponding to each helical latch.

2.3 Output Stage

The final stage in the nanomechanical logic gate is the output stage as shown Figure 1 and Figure 4. It serves a means of communicating the value computed in the logic stage to the following device or level of logic. The output stage consist of two discs; one which is stationary and the other which rotates. The stationary housing contains the output rod which, like the other stages, translates down to depict a value of 1, and no translation depicts a 0. The preset value of the output rod is 0. This value is retained between the output phase of each cycle using a compression spring. If the latched value of the helical latch is 0 as the helical latch sweeps past the output rod than no contact is made between the two resulting in an output of 0. If the helical latch has a latched value of 1 however, then the bottom tip of the helical latch rod makes contact with the ramp of the output rod. This results in a translation down by the output rod which relays the value of 1 to the output line.

The stationary housing also houses a set of rigid ramps called momentum reversing ramps (MRR). These ramps are used as a check in the system to ensure the momentum of the helical latch does not cause a false value to appear when sampling inputs. The MRR on the output stage reverses the direction of momentum of the helical latch if the compression spring of the helical latch does not react in sufficient time to latch the current value. The MRR's are located such that they make contact with the helical latch soon after the helical latch passes the ramp on the input rod. The output stage houses one MRR for each input rod.

The output rod also requires a reset mechanism in addition to its compression spring as shown in Figure 1 and Figure 4. This is achieved by using a MRR which is housed on a rotor beneath the stationary housing of the output stage. The premise is the same as with the other MRR's. It is assumed that obstruction of the tip of output rod is not permitted since it is reserved for use by the output device. Therefore the MRR is designed to interface to the side of the rod by means of a protruding lip. The MRR for the output rod could be incorporated into the bottom surface of the logic rotor, which would eliminate the output rotor, but would increase the complexity of the MRR. For simplicity to both the diagrams and the mechanism the MRR is presented with its own rotor.

3 Three-Phase Operation

The performance of the logic gate may be described temporally by a single cycle comprising 3 phases. The three phases, in order, are the input phase, the logic phase and the output phase. Logically, the primary stage for the operation of each phase has the same name, however, interaction between stages must occur in each phase. As an example, we will describe the operation of an XOR gate from its input phase to its output phase.

3.1 Input Actuation

The gate consists of a single bit output and will operate on 3 inputs as shown in Figure 1. For an XOR implementation, the function being performed can be summarized by saying that the helical latch will toggle to its opposite state with each input of 1. This requires that the latch initially be reset to a 0 state. Therefore, the input phase begins with a reset of the helical latch as shown in Figure 2. This function is performed by implementing use of the reset ramp on the input stage and the reset pin attached to the helical latch. If the helical latch is in a 1 state as it sweeps past the stationary reset ramp then contact is made and the pin rides up the ramp, causing the helical latch to translate, rotate and latch into a 0 state. If the helical latch is in the 0 state as it sweeps past the reset ramp then no contact is made and the helical latch retains its 0 state. If a reset value of 1 is desired, as with the implementation of an AND gate, then the reset pin need only be assembled on the helical latch 90 degrees out of phase with respect to that of the XOR helical latch. This results in contact with the reset ramp upon encountering a latched value of 0 and no contact with a latched value of 1. The input rods may be driven in conjunction with the resetting of the helical latch while their values are held until they have been operated on by the logic stage.

3.2 Computation of XOR

The logic phase overlaps and follows the input phase as shown in Figure 3. This is the only phase which differs significantly according to the function to be performed. The logic phase is initiated as the helical latch approaches the first input rod. When the tip of the helical latch passes beneath an input rod, one of two actions take place. If the ramp on the input rod is translated down towards the helical latch (1 state), then contact occurs between the two components. The drive force on the input rod is sufficient enough that the interaction does not effect the input rod's position. This causes the

helical latch rod to translate down the full height of the ramp. Since the latching pin contacts the helical surface similarly, to that of the reset phase, a rotation is also imposed on the rod. When the helical latch rod drops off the ramp, the compression spring and momentum reversing ramp (MRR) on the output stage latch and retain the helical latch in its new state of 1. This translation back towards the input stage of the helical latch is a fraction of the distance it traversed axially up the ramp of the input rod. After an input has been sampled, the drive force on that input is removed, and the MRR (housed on the logic rotor) ensures the input rod is reset to a 0 prior to the next cycle. The other case encountered is when the helical latch sweeps beneath an input rod having a value of 0. It is desired that the helical latch not assume a new state, therefore, no contact between the helical latch rod and input ramp occurs, and the helical latch state remains unchanged.

The logic rotor continues to sweep the helical latch to the next input. If a 1 is encountered, contact is again made. Translation, rotation and latching ensues as before with the exception that the helical surfaces now allow the latch pin to drop back to the 0 state. These transitions continue until all inputs have been sampled, at which time the logic phase of the cycle is complete. The input phase is considered to be completed when the final MRR on the logic phase resets the final input rod.

3.3 Output Generation

The output phase simply consists of passing the final value latched into the helical latch to the output rod as shown in Figure 4. As the helical latch sweeps past the output stage with a latched value of 1, then the ramp on the output rod makes contact with the tip of the helical latch and causes the output rod to translate down. This is recognized as passing a value of 1 to the following stage of logic or output device. If the last value latched into the helical latch is a 0, then no contact between the helical latch and the output rod is made and the preset value of 0 is passed. This concludes a complete cycle of logic and brings the helical latch back to the reset position; ready to operate on a new set of inputs.

As is with all Boolean logic operations for the proposed nanomechanical logic gates, the input stage and output stage geometry does not need to be affected significantly from one implementation to another. The logic stage is the functional stage and requires alteration depending upon the function implementation.

3.4 Generalized Boolean Functions

To implement an OR gate very little needs to change from the XOR gate The reset operation is identical to that of the XOR in that its reset value is 0. With an initial input of 1 the helical latch toggles to a 1 state. Again, this is identical to the XOR gate, therefore, the rising helical surface of the helical latch can be replicated from that of the XOR. An input of 0, requires no transition of the helical latch from its current state. This is achieved by virtue that no contact between the tip of the helical latch and the input is made, and the state remains unchanged (again replicating the XOR). The only change comes with successive inputs of 1. With the XOR successive inputs of 1 toggles the helical latch to its next state. With an OR gate successive inputs of 1 cannot affect the helical latch value of 1. This can be achieved by modifying the helical contour which guides the latching rod so that it does not allow the latching rod to traverse over the lip of the V groove. With additional modifications, any Boolean function can be realized in a similar manner.

3.5 Operand Scalabilty

When implementing an XOR gate using AND and OR gates, a single input increase in problem size significantly increases the number of logic gates and input lines required. With the proposed scalable logic, to increase the problem size by an additional input only requires adding the input rod to the input stage (assuming there is sufficient space on the input stage). This feature is made possible by the fact that the inputs are sampled independently, by the same helical latch. The addition of another input becomes transparent to the other inputs. The logic and output stage components remain unchanged.

4 Performance Enhancements

4.1 Symmetrical Discs

Of primary concern to the dynamic operation of the logic rotor is how the inputs may affect the stability of the housing. Since the inputs may assume binary values which may cause interaction between the input stage and logic stage, there exists a moment on the rotor. This force, a distance away from the center of rotation, could produce a wobble in the rotor as the input rod makes contact with the helical latch. The same reaction can take place in the opposite direction in the rotor when the helical latch communicates its latched value to the output rod. To compensate for these forces on one side of the rotor we can divide the discs into halves and place what was housed on a full circle into a half circle. Following this, we duplicate that half circle and rotate it 180 degrees into the empty half. The result is a system with symmetrical forces imposed on the shaft. Each force cancels out the wobbling factor of the other force. The two helical latches are shared by each half of the circle.

This duplication of components is not without advantages to the functional properties of the computing element. An intrinsic side effect to having two outputs of the same operation is a fault detection of the system. When preferred, the two outputs can be fed to an XOR gate, or comparator, to confirm the confidence of the output value.

Another side effect which can serve advantageously to the system performance by requiring symmetrical halves is that the necessary rotation of a complete cycle is also halved. The perimeter of the rotor at the helical latch obviously must double as the components to the system are doubled, however, to double the perimeter does not require that we double the radius. This property, as we

will see later, allows the maximum potential frequencies of the system to increase with the symmetrical approach.

4.2 Operation on Words

Next in the progression of enhancements to the system is the implementation of wordwise operations. Just as the discs were divided into halves previously, the halves can then be subdivided to produce sectors which contain all the necessary components to operate on individual bits in a word in parallel. In other words, we are taking the components which were housed in a semicircle and placing them into a sector which is a fraction of that size. The bit level becomes the first and simplest step in processing elements in parallel.

For example, assume a disc has been subdivided to perform XOR on two words each containing four bits. Each sector houses all the components necessary to produce an XOR output into 45 degrees of rotation of the disc. The helical latches are shared among all the inputs and outputs. This is easily justified since all the helical latches are XOR duplicates of each other. The only feature which associates a particular helical latch with a specific bit is the sector it is in during that cycle of rotation.

As the cycle begins all the helical latches are reset in parallel, requiring an equal number of reset ramps to the number of helical latches. Next, all four bits of the first word are operated on in parallel; one by each helical latch. In a likewise fashion, the four bits of the second word are operated on in parallel. Finally the latched values in each of the helical latches is output simultaneously and the cycle is concluded.

It is easily apparent that the amount of rotation required to complete a cycle is equal to the sector size; in this case it is 45 deg. More generally one cycle equals $\frac{\pi}{b}$ where b is the number of bits in a word. This feature of fractionalizing the disc allows greater operating frequencies, for the same reasons that were stated when the discs were divided into halves.

4.3 Two Level Combinational Logic

The previously unused area of the disc can be exploited for use of parallel operations as with 2 level combinational logic. Consider the expression $A\,\overline{B}\,\overline{C} + BC = R$. The AND terms for $A\,\overline{B}\,\overline{C}$ and BC can be performed in parallel on a single disc, but isolated at different distances from the center. Each set of AND terms is generated as though it were on its own disc; each has its own input rods, reset ramp, helical latch and output rod. The design simply exploits the property that the terms can be operated on in parallel and uses empty space on a three input disc to perform the two input functions. $A\,\overline{B}\,\overline{C}$ is performed close to the outside diameter of the disc since it requires a larger arc due to a third input. The second term, BC, is operated on by a separate latch at a radius within that of the first term. The offset radius must be great enough to completely isolate the two terms. There can be no interaction between the components of these two terms in the first level of logic. The arc length of the sector must meet dynamic and geometric requirements of the system.

The resultants of the first level of logic are then output to the disc housing of the second level of logic where they are operated on by an OR gate . Since the operations are synchronized, the second level of logic may be driven by the same shaft as the first level. This assumes then that the two resultants are input to the second level of logic at a common radius.

4.4 Pipelining Inputs

Pipelining is a practice used to speedup execution of consecutive instructions by breaking the execution of individual instructions into smaller steps and then running the steps of multiple instructions in parallel. This exploits the feature that instructions are executed in a sequential manner. Pipelining does not speedup the latency, the time it takes to execute a single instruction from its inception to completion. In fact, this execution time is often slowed from the sequential implementation since all of the stages must run at the cycle time of the slowest stage. Once the pipe is filled, the throughput is equal to one output per stage cycle. This feature is what offers significant time savings over individual execution of instructions.

Recognizing that the inputs of the scalable logic gate are also sampled individually, in a sequential manner, allows pipelining to occur at the gate level. Consider pipelining of a 3-input function with a 4-bit word-size. The depth of the pipe for a three-input gate is five; one for each input, one for reset, and one for output. More generally, the depth of the pipe for any one level implementation is $n+2$, where n is equal to the number of inputs. A system requirement for pipelining is that there be one latch for each stage in the pipe. The number of reset ramps, inputs and outputs remains unchanged. The processing time, or amount of rotation required to completely execute a single function, does not change. In this case it is 45 degrees. However after four stages of the pipe are filled, an output will be generated as each latch passes the output rod. In this case there will be an output generated with each 45 degrees/5 = 9 degrees of rotation.

References

[1] Drexler, K. Eric, *Nanosystems, Molecular Machinery, Manufacturing, and Computation*, John Wiley and Sons Inc., New York, N.Y., 1992.

[2] R. Mercer, M. Ebel, and R. DeMara, "Scalable Nanomechanical Logic Gate," Third Foresight Conference on Nanotechnology, 1993.

FIGURE 1
LOGIC GATE

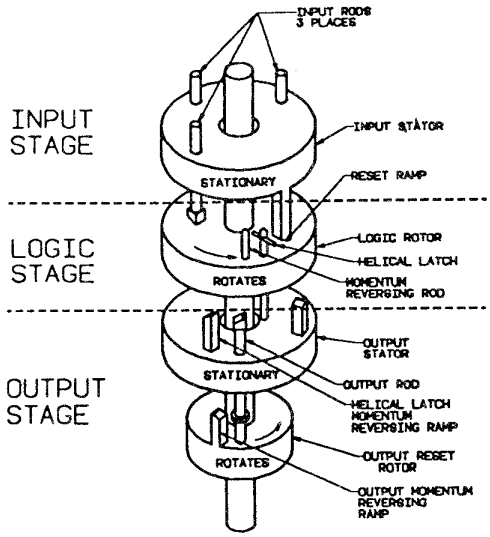

FIGURE 2
RESET OF HELICAL LATCH

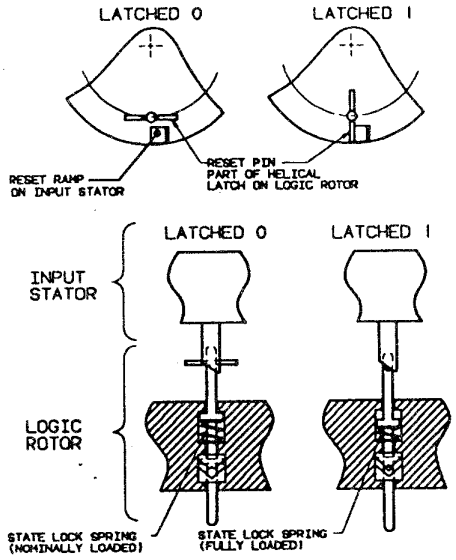

FIGURE 3
OPERATION ON SINGLE INPUT

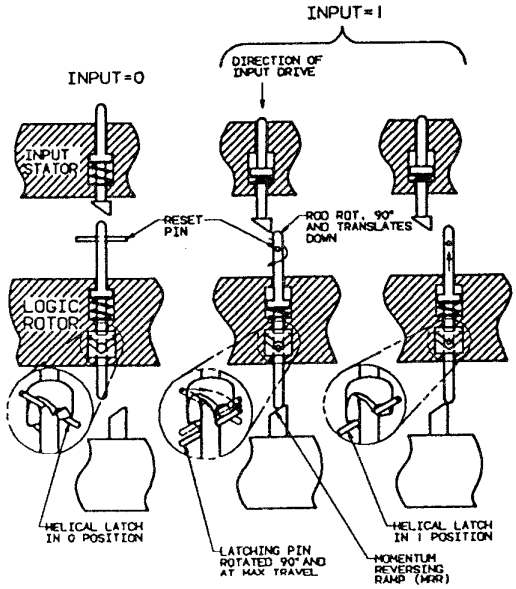

FIGURE 4
READING VALUE OF HELICAL LATCH