A multi-threshold null convention logic circuit. The circuit includes a first circuit, a first high-threshold transistor coupled to $V_{th}$, and an inverter receiving power from the first high-threshold transistor, driven by the first circuit, and including an output.

19 Claims, 19 Drawing Sheets
(56) References Cited

U.S. PATENT DOCUMENTS

7,215,155 B2 5/2007 Won
7,400,175 B2 7/2008 Fallah et al.
7,977,972 B2 7/2011 Di

OTHER PUBLICATIONS


* cited by examiner
Prior Art

Fig. 1
Prior Art

Fig. 2
FIG. 3
 Previous Partial DATA can propagate through C/L

Fig. 10C
ULTRA-LOW POWER MULTI-THRESHOLD ASYNCHRONOUS CIRCUIT DESIGN

RELATED APPLICATION


STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

This invention was made with government support under W15P77-08-C-V404 awarded by the DARPA Microsystems Technology Office. The government has certain rights in the invention.

BACKGROUND

With the current trend of semiconductor devices scaling into the deep submicron region, design challenges that were previously minor issues have now become increasingly important. Where in the past, dynamic, switching power has been the predominant factor in CMOS digital circuit power dissipation, recently, with the dramatic decrease of supply and threshold voltages, a significant growth in leakage power demands new design methodologies for digital integrated circuits (ICs). The main component of leakage power is sub-threshold leakage, caused by current flowing through a transistor even if it is supposedly turned off. Sub-threshold leakage increases exponentially with decreasing transistor feature size.

Among the many techniques proposed to control or minimize leakage power in deep submicron technology, Multi-Threshold CMOS (MTCMOS), which reduces leakage power by disconnecting the power supply from the circuit during idle (or sleep) mode while maintaining high performance in active mode, is very promising. MTCMOS incorporates transistors with two or more different threshold voltages (Vt) in a circuit. Low-Vt transistors offer fast speed but have high leakage, whereas high-Vt transistors have reduced speed but far less leakage current. MTCMOS combines these two types of transistors by utilizing low-Vt transistors for circuit switching to preserve performance and high-Vt transistors to gate the circuit power supply to significantly decrease sub-threshold leakage.

There are multiple ways to implement MTCMOS in synchronous circuits. One method is to use low-Vt transistors for critical paths to maintain high performance, while using slower high-Vt transistors for the non-critical paths to reduce leakage. Besides this path replacement methodology, there are two other architectures for implementing MTCMOS. A coarse-grained technique uses low-Vt logic for all circuit functions and gates the power to entire logic blocks with high-Vt sleep transistors, as shown in FIG. 1. The sleep transistors are controlled by a Sleep signal. During active mode, the Sleep signal is deasserted, causing both high-Vt transistors to turn on and provide a virtual power and ground to the low-Vt logic. When the circuit is idle, the Sleep signal is asserted, forcing both high-Vt transistors to turn off and disconnect power from the low-Vt logic, resulting in a very low sub-threshold leakage current. One major drawback of this method is that partitioning the circuit into appropriate logic blocks and sleep transistor sizing is difficult for large circuits. An alternative fine-grained architecture, shown in FIG. 2, incorporates the MTCMOS technique within each gate, using low-Vt transistors for the Pull-Up Network (PUN) and Pull-Down Network (PDN) and a high-Vt transistor to gate the leakage current between the two networks. Two additional low-Vt transistors are included in parallel with the PUN and PDN to maintain nearly equivalent voltage potential across these networks during sleep mode. Implementing MTCMOS within each gate solves the problems of logic block partitioning and sleep transistor sizing; however, this results in a large area overhead.

In general, three serious drawbacks hinder the widespread usage of MTCMOS in synchronous circuits: 1) the generation of Sleep signals is timing critical, often requiring complex logic circuits; 2) synchronous storage elements lose data when the power transistors are turned off during sleep mode; and 3) logic block partitioning and transistor sizing is very difficult for the coarse-grained approach, which is critical for correct circuit operation, and the fine-grained approach requires a large area overhead.

SUMMARY

The invention pertains to the fields of Computer Engineering and Electrical Engineering. The invention combines Multi-Threshold CMOS (MTCMOS) for synchronous circuits and asynchronous, delay-insensitive NULL Convention Logic (NCL), to yield an ultra-low power asynchronous circuit design methodology, named Multi-Threshold NULL Convention Logic (MTNCL).

In one embodiment, the invention provides a delay-insensitive fixed early-completion input-incomplete architecture (FECCI). The FECCI architecture includes MTNCL combinational logic blocks, which are built using MTNCL logic gates. These gates have a plurality of low-Vt transistors having high speed and high leakage current, and a plurality of high-Vt transistors having lower speed and low leakage current. The high-Vt transistors gate power to the low-Vt transistors when the MTNCL circuit is in a sleep mode. The low-Vt transistors are used for switching the output in active mode. Due to the removal of hysteresis, most MTNCL gates have fewer transistors compared to their regular NCL counterparts, thereby eliminating the area overhead. The FECCI architecture further includes a plurality of delay-insensitive early-completion registers, each register coupled to a respective MTNCL logic block to form a pipeline stage. The completion detection signal, K0, of each pipeline stage, and the handshake signal from the next stage, generate the handshake signal for the current stage, which is used as both the inverse K1 signal to the register of the current stage and the Sleep control signal of the next stage.

In another embodiment the invention provides an early-completion input-incomplete MTNCL architecture (ECII). While similar to those used in FECCI architecture, the MTNCL logic gates used in ECII architecture include hold structures, in order to avoid partial NULL from passing through since it may cause the loss of delay-insensitivity in ECII architecture.

MTNCL eliminates the three drawbacks of using the low-power MTCMOS technique with synchronous circuits (i.e., 1) the generation of Sleep signals is timing critical, often requiring complex logic circuits; 2) synchronous storage elements lose data when the power transistors are turned off during sleep mode; and 3) logic block partitioning and transistor sizing is very difficult for the coarse-grained approach, which is critical for correct circuit operation, and the fine-
grained approach requires a large area overhead), to yield an ultra-low power asynchronous circuit design methodology. MTNCL circuits have been demonstrated to dissipate 99% less static power compared to regular NCL circuits, and 38% less static power compared to MTCMOS synchronous circuits; MTNCL circuits have been demonstrated to utilize 62% less energy per operation compared to regular NCL circuits, and 99% less energy per operation compared to MTCMOS synchronous circuits; and MTNCL circuits are slightly faster than their regular NCL counterparts, and have comparable speed to synchronous circuits. Additionally, since MTNCL circuits are delay-insensitive, the supply voltage can be dropped to even sub-threshold levels to substantially further decrease power usage, at the cost of decreased performance, without requiring any circuit modifications. MTNCL circuits designed with the 1.2V IBM RF 130 nm process have been demonstrated to work correctly down to a sub-threshold supply voltage of about 0.1V, and an energy-optimal supply voltage of about 0.3V, where energy per operation is reduced by a factor of about 400x and static power is reduced by a factor of about 15x.

In another embodiment, the invention provides an early completion input incomplete asynchronous circuit (ECII). The ECII includes a delay insensitive register, a multi-threshold null convention logic circuit, and an early completion circuit. The delay insensitive register has a data input, a data output, a request input, and an acknowledge output. The multi-threshold null convention logic circuit has a data input coupled to the delay insensitive register data output, a sleep input, and an output coupled to a data input of a delay insensitive register of a subsequent ECII. The early completion circuit has a first input coupled to the acknowledge output, a second input coupled to the request input, and an output coupled to the sleep input and to a request input of a delay insensitive register of a previous ECII.

In another embodiment, the invention provides a fixed early completion input incomplete asynchronous circuit (FECCI). The FECCI includes a delay insensitive register, a multi-threshold null convention logic circuit, an early completion circuit, and an inverter. The delay insensitive register has a data input, a data output, a request input, and an acknowledge output. The multi-threshold null convention logic circuit has a data input coupled to the delay insensitive register data output, a sleep input, and an output coupled to a data input of a delay insensitive register of a subsequent FECCI. The early completion circuit has a first input coupled to the acknowledge output, a second input coupled to an output of an early completion circuit of a subsequent FECCI, and an output coupled to the sleep input and to an input of an early completion circuit of a previous FECCI. The inverter is coupled to the output of the early completion circuit and has an output coupled to the request input.

In another embodiment, the invention provides an early completion input incomplete asynchronous circuit (ECII). The circuit includes a delay insensitive register, a multi-threshold null convention logic circuit, and a sleep early completion circuit. The delay insensitive register has a data input and a data output. The multi-threshold null convention logic circuit has a data input coupled to the delay insensitive register data output, a sleep input, and a data output. The sleep early completion circuit has a first input coupled to an output of an early completion circuit of a subsequent ECII, a second input coupled to the output of an early completion circuit of a previous ECII, a third input coupled to the delay insensitive register data input, and an output coupled to the sleep input.

In another embodiment, the invention provides a multi-threshold null convention logic circuit. The circuit includes a first circuit coupled to $V_{DD}$, a set circuit coupled to the first circuit, a first high-threshold transistor coupled between the set circuit and ground, a second high-threshold transistor coupled to $V_{DD}$, a first low-threshold transistor coupled to the second high-threshold transistor, a third high-threshold transistor coupled between the first low-threshold transistor and ground, a second low-threshold transistor coupled between the coupling of the third high-threshold transistor and the first low-threshold transistor and ground, and an output coupled to the coupling between the third high-threshold transistor, the first low-threshold transistor, and the second low-threshold transistor. The first high-threshold transistor is driven by a SLEEP signal. The second high-threshold transistor is driven by a SLEEP signal. The first low-threshold transistor is driven by the coupling between the first circuit and the set circuit. The third high-threshold transistor is driven by the coupling between the first circuit and the set circuit. The second low-threshold transistor is driven by the SLEEP signal.

In another embodiment, the invention provides a multi-threshold null convention logic circuit. The circuit includes a first circuit, a first high-threshold transistor coupled to $V_{DD}$, and an inverter receiving power from the first high-threshold transistor, driven by the first circuit, and including an output.

In another embodiment, the invention provides a multi-threshold null convention logic circuit. The circuit includes an inverter coupled to ground and having an output, a first circuit driving the inverter, and a high-threshold transistor driven by a sleep signal having an asserted and a non-asserted state. The high-threshold transistor couples the inverter to $V_{DD}$ when the sleep signal is in the non-asserted state and isolates the inverter from $V_{DD}$ when the sleep signal is in the asserted state. The first circuit also drives the inverter with a logic 1 during a sleep mode when the sleep signal is asserted. The circuit prevents a glitch on the output when the sleep signal switches from the asserted state to the non-asserted state.

In another embodiment, the invention provides a method of manufacturing a logic circuit having an output and receiving a sleep signal having an asserted state and a non-asserted state. The method includes providing a first circuit for performing the logic, providing an inverter driven by the second output, and providing a sleep circuit to couple and isolate the inverter to $V_{DD}$ and the first circuit to ground based on the state of the sleep signal. The first circuit is coupled to $V_{DD}$ by one or more high-threshold transistors, and has a second output. The inverter drives the output.

Other aspects of the invention will become apparent by consideration of the detailed description and accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a schematic diagram of a prior art MTCMOS circuit architecture.

FIG. 2 is a schematic diagram of a prior art MTCMOS circuit as applied to a Boolean gate.

FIGS. 3a-3d are block diagrams of a MTNCL pipeline architecture using regular completion.

FIGS. 4a-4d are block diagrams of a MTNCL pipeline architecture using early completion.

FIGS. 5a and 5b are schematic diagrams of original MTCMOS NCL threshold gates.

FIG. 6 is a schematic diagram of an original MTCMOS static TH23 gate.

FIG. 7a is a schematic diagram of a static TH23 NCL implementation.
FIG. 7b is a schematic diagram of a semi-static TH23 NCL implementation.
FIG. 8a is a schematic diagram of an original static MT MCOS NCL threshold gate structure.
FIG. 8b is a graph showing an output glitch produced by the gate structure of FIG. 8a.
FIG. 9a is a schematic diagram of a SMTNCL gate structure.
FIG. 9b is a schematic diagram of a SMTNCL TH23 implementation.
FIGS. 10a-10c are block diagrams of an ECII architecture using SMTNCL gates.
FIG. 11 is a block diagram of a fixed early-completion input-incomplete architecture (FECH).
FIG. 12a is a schematic diagram of a SMTNCL1 gate structure.
FIG. 12b is a schematic diagram of a SMTNCL1 TH23 implementation.
FIG. 13a is a schematic diagram of a SSMTNCL1 gate structure.
FIG. 13b is a schematic diagram of a SSMTNCL1 TH23 implementation.
FIG. 14a is a schematic diagram of a SSMTNCL gate structure.
FIG. 14b is a schematic diagram of a SSMTNCL TH23 implementation.
FIG. 15 is a block diagram of a Slep Early Completion Input-Incomplete (SECI) architecture.
FIG. 16 is a schematic diagram of an Early completion component with Sleep input.
FIG. 17 is a block diagram of a Slep Early Completion and Registration Input-Incomplete (SECR) architecture.
FIG. 18 is a schematic diagram of a DJ register with Sleep input.

DETAILED DESCRIPTION

Before any embodiment of the invention are explained in detail, it is to be understood that the invention is not limited in its application to the details of construction and the arrangement of components set forth in the following description or illustrated in the following drawings. The invention is capable of other embodiments and of being practiced or of being carried out in various ways.

The three drawbacks of MT MOS are eliminated by utilizing NULL Convention Logic (NCL) in conjunction with the MT MOS technique, as described in embodiments of this invention, yielding an ultra-low power asynchronous circuit design methodology, referred to as Multi-Threshold NULL Convention Logic (MTNCL).

Early-Completion Input-Incomplete (ECII) MTNCL Architecture

NCL threshold gates are larger and implement more complicated functions than basic Boolean gates, such that fewer threshold gates are normally needed to implement an arbitrary function compared to the number of Boolean gates; however, the NCL implementation often requires more transistors. Therefore, incorporating MT MOS inside each threshold gate will likely facilitate easy sleep transistor sizing without requiring as large an area overhead. Since floating nodes may result in substantial short circuit power consumption at the following stage, an MT MOS structure similar to the one shown in FIG. 2 is used to pull the output node to ground during sleep mode. All of the MTNCL gates in a pipeline stage being in sleep mode (i.e., all gate outputs are logic 0) is equivalent to the pipeline stage being in the NULL state. Hence, after each DATA cycle, all MTNCL gates in a pipeline stage can be forced to output logic 0. By asserting the sleep control signal, instead of propagating a NULL waveform through the stage, data is not lost during sleep mode.

Since a complete detection signal, Ko, indicates whether the corresponding pipeline stage is ready to undergo a DATA or NULL cycle, Ko can be used as a sleep control signal without requiring any additional hardware. In contrast, complex Sleep signal generation circuitry is needed for synchronous MT MOS circuits. However, directly implementing this idea, using regular NCL completion, compromises delay-insensitivity.

Referring to FIG. 3, each inverted completion signal is used as the sleep signal for all MTNCL gates in the corresponding pipeline stage. Looking at the left stage, after a DATA (D) cycle, the completion signal becomes request for NULL (rfl), (i.e., logic 0), which forces all threshold gates in the stage to enter sleep mode, since the next cycle will be NULL (N). When this sleep generated NULL waveform is latched by the subsequent register, the stage’s completion signal switches back to request for data (rd) (i.e., logic 1). If this occurs before all bits of the preceding DATA waveform become NULL, the non-NULL preceding waveform bits will be retained and utilized in the subsequent operation, thereby compromising delay-insensitivity.

Early-completion can be used in lieu of regular completion to solve this problem. With early-completion, the register inputs are used, instead of the outputs, along with the previous stage’s completion signal to generate the current stage’s completion signal. As shown in FIG. 4, each completion signal is used as the sleep signal for all threshold gates in the subsequent pipeline stage. Now the combinational logic won’t be put to sleep until all inputs are NULL and the stage is requesting NULL; therefore the NULL waveform is ready to propagate through the stage, so the stage can be put to sleep without compromising delay-insensitivity. The stage will then remain in sleep mode until all inputs are DATA, and the stage is requesting DATA (i.e., the stage is ready to evaluate). The early-completion MTNCL architecture ensures input-completeness through the sleep mechanism (i.e., the circuit is only put to sleep after all inputs are NULL, and only evaluates after all inputs are DATA). Using input-incomplete logic functions to design the circuit decreases area and power and increases speed.

MTNCL Threshold Gate Design for ECII Architecture

Incorporating MT MOS structure inside each NCL threshold gate, results in a number of the original transistors no longer being needed. For example, as shown in FIG. 5a, the reset circuitry is no longer needed, because the gate output is forced to NULL by the MT MOS sleep mechanism (instead of all inputs becoming logic 0). Hold1 was used to ensure that the gate remains asserted, once it has become asserted, until all inputs are deasserted. This guarantees input-completeness with respect to the NULL waveform. However, since the ECII architecture guarantees input-completeness through the sleep mechanism, NCL gate hysteresis is no longer required. Hence, the hold1 circuitry, and corresponding NMOS transistor, are removed. The PMOS transistor is also removed to maintain the complementary nature of CMOS logic (i.e., set and hold0 are complements of each other), such that the gate does not float.

A direct MT MOS NCL threshold gate implementation, similar to the structure shown in FIG. 2, is shown in FIG. 5b. All PMOS transistors, except the inverter, are high-Vt, (denoted by a dotted circle). This is possible because they are only turned on when the gate enters sleep mode and the inputs become logic 0, and they remain on when the gate exits sleep mode, until the gate’s set condition becomes true. In both
cases, the gate output is already logic 0; therefore, the speed of these PMOS transistors does not affect performance, and high-Vt transistors are used to reduce leakage current. During active mode, the Sleep signal is logic 0 and Sleep is logic 1, such that sleep transistors M1 and M2 are turned on, and bypass transistors M3 and M4 and the output pull-down transistor M5 are turned off, and the gate functions as normal. During sleep mode, Sleep is logic 1 and Sleep is logic 0, such that M5, which is a low-Vt transistor, is turned on, quickly pulling the output to logic 0, while M3 and M4 are turned on to minimize the voltage across the hold0 and set blocks, respectively. The high-Vt gating transistors, M1 and M2, are turned off to reduce leakage. An example of an MTNCL implementation of a static TH23 gate is shown in FIG. 6, whereas an original static TH23 gate is shown in FIG. 7a.

Note that the MTNCL TH23 gate is actually smaller than the original TH23 gate (i.e., 17 vs. 18 transistors). Although the MTNCL structure adds five transistors to each gate, this only increases the total number of transistors for any of the 32 threshold gates, since the reset and hold1 blocks are removed. This is true in the total number of transistors in the MTNCL version normally being less than the original version. Exceptions include the two TH1n gates, which are OR gates, and therefore do not have an extra hysteretic circuit (i.e., reset=hold0 and set=hold1), no transistors are removed for the MTNCL versions. Therefore, these three MTNCL gates require 5 additional transistors each.

This initial MTNCL static threshold gate structure has been used to implement an 8-bit/8-bit pipeline array multiplier using a 1.2V 130 nm IBM SRF CMOS process, reducing leakage power 150 times and reducing active energy by 1.8 times compared to a regular NCL low-Vt counterpart. However, as shown in FIG. 8b, this structure produces unwanted glitches at the gate outputs. Referring to FIG. 8a, during sleep mode, Sleep is logic 1 and Sleep is logic 0; Q1 and Q5 are off, while Q1, Q2, and Q4 are on. The internal parasitic capacitance, C delighted by C1 and C3, makes the internal node, p, logic 0. When the gate is taken out of sleep mode, Sleep is logic 0 and Sleep is logic 1, Q1 and Q5 are on, and Q2, Q3, and Q4 are off. Since all inputs are logic 0 at this moment, due to the preceding NULL sleep cycle, C4 begins charging through the PMOS network and Q1. However, before the voltage on p rises to VTH(1-VTH) (where VTH is the threshold voltage of the PMOS transistor in the output inverter), the gate output (i.e., the input signal to the inverter and the voltage on C4) starts to rise, tauto खड़े on the PMOS transistor, and causing a glitch shown in FIG. 8b. With a supply voltage of 1.2V, these glitches can be as high as 400 mV, and are able to propagate through logic gates. Although a multiplier test circuit still functioned correctly, these glitches need to be removed to ensure reliable operation and eliminate glitch power.

To reduce or eliminate the glitch, the MTNCL static threshold gate structure is modified, as shown in FIG. 9, by moving the power gating high-Vt transistor to the PDN such that during sleep mode the internal node is charged to logic 1. The two bypass transistors, Q3 and Q4, which have only a very minimal contribution to leakage savings can be removed to reduce the area of the gate. Therefore, when the gate is taken out of sleep mode, the output will remain at logic 0 without any glitch, due to the internal logic 1 flowing through the output inverter, until the DATA wavefront arrives. Since the internal node is logic 1 during sleep mode, and the output is logic 0, the NMOS transistor in the output inverter is no longer on the critical path and can be a high-Vt transistor. This modified Static MTNCL threshold gate structure is referred to as SMTNCL.

Delay-Insensitivity Analysis

Combining the ECI architecture with the SMTNCL gate structure results in a delay-insensitivity problem after a DATA cycle, if most, but not all, inputs become NULL (see FIG. 10). Because the subsequent stage is requesting NULL, and because the hold 1 logic has been removed from the SMTNCL gates, this Partial NULL (PN) wavefront can pass through the stage's input register causing all stage outputs to become NULL, before all inputs are NULL and the stage is put to sleep. This violates an input-completeness criteria that requires that all outputs of a combinational circuit to not transition from NULL to DATA until all inputs have transitioned from NULL to DATA, and that all outputs of a combinational circuit may not transition from DATA to NULL until all inputs have transitioned from DATA to NULL. This can cause the subsequent stage to request the next DATA while the previous stage input is still a partial NULL, resulting in the preceding wavefront bits that are still DATA being retained and utilized in the subsequent operation. This can compromise delay-insensitivity, similar to the problem encountered using regular completion, as explained above.

There are at least two solutions to this problem, one at the architecture level and the other at the gate level. Since the problem is caused by a partial NULL passing through the register, this can be fixed at the architecture level by ensuring that the NULL wavefront is only allowed to pass through the register after all register inputs are NULL. This is achievable by a Fixed Early-Completion Input-Incomplete (FECC) architecture which uses the stage's inverted sleep signal as its input register's ki signal. The FECC architecture is shown in FIG. 11. Compared to ECI, FECC is slower because the registers must wait until all inputs become DATA/NULL before they are latched. Note that because the stage will remain in sleep mode until all inputs are DATA, a partial DATA wavefront passing through the register does not pose a problem, and all stage outputs will remain NULL until all inputs are DATA.

This problem can be solved at the gate level using a Static MTNCL implementation with hold1 (SMTNCL1), which requires 2 additional transistors vs. the SMTNCL gate. The feedback NMOS transistor is sufficient to hold the output at logic 1, without requiring the traditional NCL hold1 circuitry, because this ensures that once the gate output has been asserted due to the current DATA wavefront, that it will only be deasserted when the gate is put to sleep (i.e., when all circuit inputs are NULL), and will not be reasserted due to a partial NULL wavefront. The SMTNCL1 design is shown in FIG. 12.

The MTMOS structure can also be applied to semi-static NCL gates, which utilize a weak feedback inverter to implement the hold1 and hold0 functions. This Semi-Static MTNCL design with hold1 (SSMTNCL1), is shown in FIG. 13. Note that a 1 (one) at the end of an MTNCL gate name denotes that the gate includes hold1 circuitry. The NMOS transistor in the weak inverter that serves as the hold1 function is not needed for the FECC architecture; and can be removed to save area. This results in a modified Semi-Static MTNCL design (SSMTNCL) shown in FIG. 14.

To summarize, the ECI architecture works with SMTNCL1 or SSMTNCL1 gates, which include the hold1 function. The FECC architecture works with all four MTNCL gate designs (i.e., SMTNCL1, SSMTNCL1, and SSMTNCL), however, the SMTNCL and SSMTNCL gates require fewer transistors than their equivalent SMTNCL and
SSMTNCL gates, such that the FECII architecture would normally use either the SMTNCL or SSMTNCL gates. Additionally, the ECII architecture is faster than FECII, and the static NCL gates (i.e., SMTNCL and SSMTNCL) perform better than their semi-static counterparts (i.e., SSMTNCL and SSMTNCL) at reduced supply voltage. This is because the static implementations do not utilize a weak inverter, which ceases to operate properly with a substantially reduced supply voltage, for state-holding.

In the MTNCL architectures described so far, an NCL circuit's C/L was split in lieu of the NULL cycle to significantly reduce leakage power. However, during sleep mode the circuit's completion and registration logic remains active, which for a fine grain pipelined circuit may be a significant portion of the logic. Therefore, it would be very beneficial to be able to sleep the completion and registration logic in addition to the C/L. The completion logic can be split by modifying the ECII architecture, shown in FIG. 4, to include a sleep input to the completion logic and use SMTNCL gates to implement the completion logic, as shown in FIGS. 15 and 16, respectively. Note that the final inverting TH22 gate is a regular NCL gate, which is not split. This is consistent with the NULL cycle, where the internal completion component gates are all logic 0, except for the final inverting TH22 gate.

During a NULL cycle, the register output is also NULL, so it can be slept, as shown in FIG. 17. Instead of using two SMTNCL TH22 gates to implement the register, the sleep transistors for each rail can be combined, such that a dual-rail register is implemented as a single component to reduce area, as shown in FIG. 18. Note that this architecture is similar to the FECII architecture shown in FIG. 11, which does not allow a partial NULL to propagate through the register, such that the C/L can be implemented with smaller SMTNCL gates instead of SMTNCL gates.

Table I compares the various MTNCL architectures and original NCL circuits using a 4-stage pipelined IEEE single-precision floating-point co-processor, which performs addition, subtraction, and multiplication.

<table>
<thead>
<tr>
<th></th>
<th># Transistors</th>
<th>$\tau_{90}$ (ms)</th>
<th>Energy/Operation (pJ)</th>
<th>Leakage Power ($\mu$W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCL Low Vcc</td>
<td>158059</td>
<td>14.1</td>
<td>25.6</td>
<td>13969</td>
</tr>
<tr>
<td>NCL High Vcc</td>
<td>158059</td>
<td>32.5</td>
<td>25.9</td>
<td>300</td>
</tr>
<tr>
<td>SMTNCL with FECII</td>
<td>111506</td>
<td>13.4</td>
<td>20.7</td>
<td>377</td>
</tr>
<tr>
<td>SMTNCL with ECII</td>
<td>119706</td>
<td>14.3</td>
<td>20.6</td>
<td>382</td>
</tr>
<tr>
<td>SMTNCL with SECII</td>
<td>110244</td>
<td>12.8</td>
<td>20.5</td>
<td>382</td>
</tr>
<tr>
<td>SMTNCL with SECII</td>
<td>96640</td>
<td>12.6</td>
<td>19.5</td>
<td>346</td>
</tr>
</tbody>
</table>


What is claimed is:

1. A multi-threshold null convention logic circuit, the circuit comprising:
a first circuit;
a first high-threshold transistor coupled to $V_{cc}$; and
an inverter receiving power from the first high-threshold transistor, driven by the first circuit, and including an output;
wherein the set circuit is isolated from ground during a sleep cycle.

2. The circuit of claim 1, wherein the first high-threshold transistor is driven by a sleep signal.

3. The circuit of claim 1, wherein a second high-threshold transistor couples the first circuit to ground and is driven by a sleep signal.

4. The circuit of claim 1, wherein a first low-threshold transistor couples the inverter output to ground and is driven by a sleep signal.

5. The circuit of claim 1, wherein the output is driven to ground during a sleep cycle.

6. The circuit of claim 1, wherein the first circuit includes a hold0 circuit and a set circuit coupled together.

7. The circuit of claim 1, wherein the first circuit drives the inverter with a logic 1 during a sleep mode following assertion of the sleep signal.

8. The circuit of claim 7, wherein isolating the inverter from $V_{cc}$ and driving the inverter with a logic 1 during the sleep cycle prevents a glitch upon exiting the sleep cycle.

9. The circuit of claim 1, wherein the inverter is isolated from $V_{cc}$ during a sleep cycle.

10. The circuit of claim 1, wherein the first circuit is coupled to $V_{cc}$ exclusively by one or more high-threshold transistors.

11. A multi-threshold null convention logic circuit, the circuit comprising:
an inverter coupled to ground and having an output;
a first circuit driving the inverter; and
a high-threshold transistor driven by a sleep signal having an asserted and a non-asserted state, and coupling the inverter to $V_{cc}$ when the sleep signal is in the non-asserted state and isolating the inverter from $V_{cc}$ when the sleep signal is in the asserted state, and
a second high-threshold transistor driven by a second sleep signal having a second asserted state and a second non-asserted state, the second high-threshold transistor coupling the first circuit to ground when the second sleep signal is in the second asserted state and isolating the first circuit from ground when the second sleep signal is in the second non-asserted state;
wherein the first circuit drives the inverter with a logic 1 during a sleep mode following assertion of the sleep signal;
wherein a glitch on the output when the sleep signal switches from the asserted state to the non-asserted state is prevented.

12. The circuit of claim 11, wherein the first circuit includes a plurality of high-threshold transistors and low-threshold transistors, the first circuit coupled to $V_{cc}$ by only high-threshold transistors.

13. The circuit of claim 11, wherein the second signal is in the second non-asserted state when the sleep signal is in the asserted state and is in the second asserted state when the sleep signal is in the non-asserted state.

14. A method of manufacturing a logic circuit having an output and receiving a sleep signal having an asserted state and a non-asserted state, the method comprising:

provide a first circuit for performing the logic, the first circuit coupled to $V_{cc}$ by one or more high-threshold transistors, the first circuit having a second output;

providing an inverter driven by the second output, the inverter driving the output; and

providing a sleep circuit to couple and isolate the inverter to $V_{cc}$ and the first circuit to ground based on the state of the sleep signal; and

isolating the first circuit from ground via the high-threshold transistor of the sleep circuit when the sleep signal is in the asserted state.
15. The method of claim 14, further comprising coupling the inverter to Vcc via a high-threshold transistor of the sleep circuit when the sleep signal is in the non-asserted state.

16. The method of claim 14, further comprising isolating the inverter from Vcc via the high-threshold transistor of the sleep circuit when the sleep signal is in the asserted state.

17. The method of claim 14, further comprising coupling the first circuit to ground via a high-threshold transistor of the sleep circuit when the sleep signal is in the non-asserted state.

18. The method of claim 14, further comprising coupling the output to ground via a low-threshold transistor when the sleep signal is in the asserted state.

19. The method of claim 14, wherein the logic circuit is a threshold gate.