

High-Speed Energy Estimation for Delay-Insensitive Circuits

Bonita Bhaskaran, Venkat Satagopan, and Scott C. Smith
University of Missouri – Rolla, Department of Electrical and Computer Engineering
133 Emerson Electric Co. Hall, 1870 Miner Circle, Rolla, MO 65409
Phone: (573) 341-4232, Fax: (573) 341-4532, E-mail: smithsco@umr.edu

Keywords: Asynchronous circuits, delay-insensitive circuits, energy, power, NULL Convention Logic (NCL)

Abstract

With increasingly smaller feature sizes and higher on-chip densities, the power dissipation of VLSI systems has become a primary concern for designers. This paper presents a fast and efficient energy estimation approach for delay-insensitive (DI) systems, based on gate-level switching. The approach has been automated and works with standard industrial design tool suites, such as Mentor Graphics and Synopsys. The method is applied to the NULL Convention Logic (NCL) DI paradigm, and is tested on a number of different NCL multiplier architectures. The results from the developed gate-level switching method are compared to those from transistor-level simulation, showing that the method developed herein produces results more than 1000 times as fast, that fall within the result range obtained by two different industry-standard transistor-level simulators, for the tested designs. This method is extremely useful for quickly determining how architecture changes will affect energy usage.

1. Introduction

For the last two decades the focus of digital design has been primarily on synchronous, clocked architectures. However, as clock rates have significantly increased while feature size has decreased, clock skew and fabrication process variations have become major problems. High performance chips must dedicate increasingly larger portions of their area for clock drivers to achieve acceptable skew, assuming normal (expected) fabrication process variations, causing these chips to dissipate increasingly higher power, especially at the clock edge, when switching is most prevalent. As these trends continue, the clock is becoming more and more difficult to manage. This has caused renewed interest in asynchronous digital design. As the demand continues for

higher performance, higher complexity, and decreased feature size, asynchronous paradigms will become more widely used in the industry. The International Technology Roadmap for Semiconductors predicts a likely shift from synchronous to asynchronous design styles in order to increase circuit robustness, decrease power, and alleviate many clock-related issues [1].

Delay-insensitive asynchronous paradigms, like NULL Convention Logic (NCL) [2], have numerous advantages over their clocked Boolean counterparts, including reduced timing effort, power, noise, and EMI, increased robustness and design reusability, and suitability for System-on-Chip (SoC) design. Delay-insensitivity also yields average-case, versus worst-case performance, no glitch power, and distributes the demand for power over time and area, reducing the occurrence of hot spots and peak power demand. However, asynchronous circuits haven't yet gained widespread industrial popularity due mostly to the lack of industry-standard CAD tool support.

Since low power is one of the main advantages of asynchronous systems, power/energy usage is an important benchmark for delay-insensitive (DI) circuits. Traditional transistor-level simulators are quite powerful and yield very accurate results; however, they require an extremely long run time even for moderately sized circuits. DI circuits do not rely on a global clock tree network; instead they use local handshaking signals for synchronization. Furthermore, they only consume energy when performing useful work; the rest of the time no switching occurs, so energy consumption is nominal, when using CMOS gates. It is therefore feasible to utilize gate-level switching instead of transistor-level switching for estimating the energy consumption of DI circuits.

2. NCL Overview

NCL is a self-timed logic paradigm in which control is inherent in each datum. NCL follows the so-called weak

conditions of Seitz's delay-insensitive signaling scheme [3]. Like other delay-insensitive logic methods, the NCL paradigm assumes that forks in wires are isochronic [4]. Various aspects of the paradigm, including the NULL (or spacer) logic state from which NCL derives its name, have origins in Muller's work on speed-independent circuits in the 1950s and 1960s [5].

2.1 Delay-Insensitivity

NCL uses symbolic completeness of expression to achieve delay-insensitive behavior. A symbolically complete expression depends only on the relationships of the symbols present in the expression without reference to their time of evaluation [2]. In particular, dual-rail and quad-rail signals, or other mutually exclusive assertion groups (MEAGs), can incorporate data and control information into one mixed control/data path to eliminate time reference. For NCL, and other DI paradigms, to be purely delay-insensitive, assuming isochronic wire forks [4], they must meet the input-completeness and observability criteria [6].

Completeness of input requires that all the outputs of a combinational circuit may not transition from NULL to DATA until all inputs have transitioned from NULL to DATA, and that all the outputs of a combinational circuit may not transition from DATA to NULL until all inputs have transitioned from DATA to NULL. In circuits with multiple outputs, it is acceptable, according to Seitz's weak conditions [3], for some of the outputs to transition without having a complete input set present, as long as all outputs cannot transition before all inputs arrive. Observability requires that no *orphans* may propagate through a gate [7]. An orphan is defined as a wire that transitions during the current DATA wavefront, but is not used in the determination of the output. Orphans are caused by wire forks and can be neglected through the isochronic fork assumption [4], as long as they are not allowed to cross a gate boundary. This *observability* condition, also referred to as *indicatability* or *stability*, ensures that every gate transition is observable at the output; which means that every gate that transitions is necessary to transition at least one of the outputs. Furthermore, when circuits use the bit-wise completion strategy with selective input-incomplete components, they must also adhere to the completion-completeness criterion [8], which requires that completion signals only be generated such that no two adjacent DATA wavefronts can interact within any combinational component.

Most multi-rail delay-insensitive systems, including NCL, have at least two register stages, one at both the input and the output. Two adjacent register stages interact through request and acknowledge lines, K_i and K_o , to prevent the current DATA wavefront from overwriting

the previous DATA wavefront by ensuring that the two are always separated by a NULL wavefront.

2.2 Logic Gates

NCL differs from other delay-insensitive paradigms, like [3], which use only one type of state-holding gate, the C-element [5]. A C-element behaves as follows: when all inputs assume the same value, the output assumes this value; otherwise, the output does not change. On the other hand, all NCL gates are state-holding. NCL uses threshold gates as its basic logic elements [9]. The primary type of threshold gate is the TH_{mn} gate ($1 \leq m \leq n$). TH_{mn} gates have n inputs. At least m of the n inputs must be asserted before the output becomes asserted. Because NCL threshold gates are designed with *hysteresis*, all asserted inputs must be deasserted before the output is deasserted. Hysteresis ensures a complete transition of inputs back to NULL before asserting the output associated with the next wavefront of input data. NCL threshold gates may also include a *reset* input to initialize the output. Circuit diagrams designate resettable gates by either a D or an N appearing inside the gate, along with the gate's threshold. D denotes the gate as being reset to logic 1; N , to logic 0.

3. Previous Work

Since low power is one of the main advantages of asynchronous systems, power/energy usage is an important benchmark for DI circuits. Previous methods use transistor-level simulation for precise calculation of power or energy consumption. This can be done with Cadence, Powermill, or Mentor Graphics tools, among others. The following example pertains to using the Mentor Graphics tools for power/energy calculation. First, a structural VHDL description of the circuit is transformed to an EDIF netlist using the synthesis tool, Leonardo Spectrum. This file consists solely of threshold gates, which are in the form of black boxes. Next, the EDIF 200 Netlister tool is used to map the black box threshold gates to their equivalent transistor-level circuits. Schematic Generator is then used to create the transistor level circuit diagram. Using Design Architect, the transistor-level circuit diagram is opened and its viewpoint is created. Finally, the transistor-level circuit is simulated using Accusim II, an analog circuit simulator, which yields the necessary voltage and current waveforms used to calculate power and energy, using the Waveform Calculator.

This process is extensive and the simulation time is very long. Another major drawback to this technique is that the designer needs to manually create and specify force files for the circuit inputs. This is a well-established method for synchronous circuits, and is easy to use when

the inputs change relative to a periodic clock pulse. However, the inputs to DI systems change in response to handshaking outputs, which are not periodic. Therefore, changing the inputs at regular intervals does not work well. To alleviate this problem, we developed an alternative method for simulating transistor-level circuits that uses a VHDL testbench to provide the transistor-level circuit inputs, based on the transistor-level circuit outputs. Furthermore, it allows for automatic checking of functional correctness, since the transistor-level circuit outputs are read by the VHDL testbench. This method, as detailed in [10], uses the Mentor Graphics Advance MS tool (ADMS), and works equally well for synchronous circuits. However, the simulation time for this method is also extremely long, even for moderately sized circuits, like a 4-bit \times 4-bit multiplier, which takes approximately 24 hours to run an exhaustive 256-input combination test on a 900 MHz Sun machine.

4. Energy Metric

There are three main contributing factors to power dissipation in CMOS VLSI systems: dynamic, short-circuit, and static power. Dynamic dissipation results whenever there is a transition in the outputs due to the charging and discharging of capacitance. Short-circuit dissipation occurs when the p-type and n-type networks are both momentarily ON, leading to a direct connection path between V_{dd} and *ground*. This also occurs during output transitions. Static dissipation is due to leakage current in the transistors when they are not switching. Ideally, CMOS circuits have very little leakage current; however, the leakage currents do become significant for very high density chips. In DI CMOS circuits, static power dissipation is negligible compared to the dynamic and short-circuit power; therefore, only the switching power needs to be taken into consideration when estimating energy consumption. Hence, energy is only used when a gate transitions.

Energy consumption and power dissipation are two metrics of assessing performance of VLSI systems and are closely related to each other. Power is an instantaneous value, and is defined as the source voltage, V_{dd} , multiplied by the current through V_{dd} , $I_{V_{dd}}$. Average power can then be calculated by plotting $I_{V_{dd}} \times V_{dd}$ and finding the average value of this waveform. The total energy is equal to the area under the power waveform, $\int (I_{V_{dd}} \times V_{dd})$.

A popular metric in synchronous circuits is *energy per clock cycle*. This is calculated by dividing the total energy for the simulation by the number of clock cycles in the simulation. Since there is no clock in DI circuits, a meaningful metric is *energy per operation* [11] (i.e. how much energy does it take to do one multiplication operation on average). The metric is a direct measure of

the amount of work done per operation, providing a measure of how battery life will be affected, and is good for determining the effects that different architectures have on energy usage. A comparable value for synchronous circuits can be calculated by multiplying the energy per clock cycle by the average number of clock cycles per operation, to obtain energy per operation. The equation for energy per operation is given below.

$$E = \frac{\int_0^t I_{V_{DD}}(t) \times V_{DD} dt}{\text{number of operations}} \text{ Joules} \quad (1)$$

5. Energy Estimation Technique

Figure 1 shows a transistor level simulation of an NCL TH33 gate, and its corresponding energy waveform, using Mentor Graphics Accusim II tool. There are a few key points to note that make our gate-level switching approach viable for NCL DI systems:

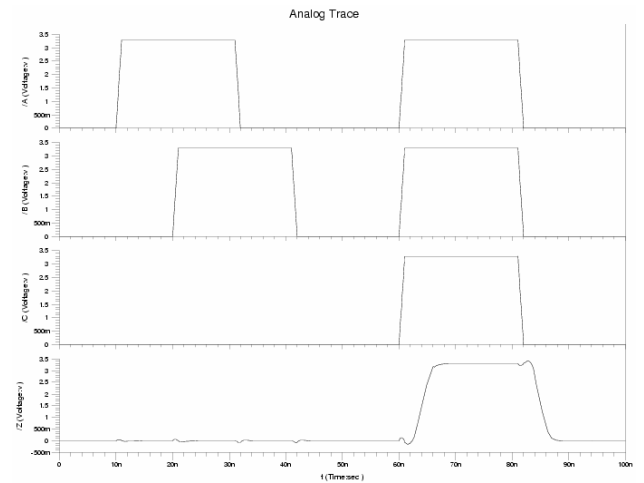


Figure 1a. TH33 gate I/O waveforms.

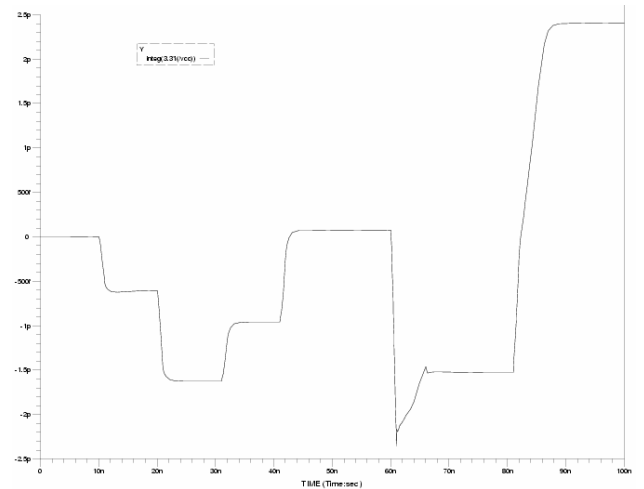


Figure 1b. TH33 gate energy waveform.

- 1) Inputs continuously alternate between DATA and NULL, such that all wires asserted in a DATA wavefront are always deasserted in the corresponding NULL wavefront. Figure 1 shows that NCL gates that do not transition during a DATA wavefront utilize an insignificant amount of energy, which can be ignored. Specifically, from 0-50 ns, two inputs of the TH33 gate are asserted during the DATA wavefront, but the 3rd is not asserted, so the output is not asserted. The inputs are then deasserted during the NULL wavefront. Figure 1b shows that these transitions only require 0.07 pJ, whereas approximately 33 times more energy (2.34 pJ) is required when the gate switches; hence, the non-switching energy is insignificant and can be ignored.
- 2) Figure 1b shows that the static power is indeed negligible, since the energy only changes when the inputs change, and remains relatively constant otherwise; hence, this static power can also be ignored.
- 3) Since inputs continuously alternate between DATA and NULL, such that all wires asserted in a DATA wavefront are always deasserted in the corresponding NULL wavefront, all gates asserted in the DATA wavefront will be deasserted in the NULL wavefront. Therefore, the total energy for a gate to transition (0→1→0) can be added on the 0→1 transition, with no energy added for the 1→0 transition.
- 4) Also note that the order and timing in which the inputs transition does not significantly change the amount of energy used in the gate transition (i.e. for the TH33 gate, asserting *A*, then *B*, then *C*, requires approximately the same amount of energy as when *A*, *B*, and *C* are all asserted simultaneously).
- 5) However, for gates with multiple set conditions, the specific asserted input set causing the gate to become asserted does significantly affect the amount of energy required to assert the gate's output. Hence, multiple energies corresponding to each set condition must be used by the estimation tool for each gate. Take for example a TH23 gate. The transition due to *AB* requires 5.85 pJ; the one due to *AC* requires 5.07 pJ; and the transition due to *BC* requires 3.83 pJ. Note that asserting the 3rd input once the gate is already asserted does not require any significant energy; hence, this transition can be ignored.
- 6) Finally, note that NCL systems adhere to monotonic transitions between DATA and NULL, which means that there are no glitches. This is important because glitches can require a significant amount of energy, and would be hard to incorporate into the tool using the proposed method because they could lead to gates only partially transitioning (i.e. a gate's output may start to rise, but if the inputs change, the output may then fall back to zero, or vice versa, which requires a

significant amount of energy, even though the gate's output did not switch).

The energy estimation technique developed herein is based on gate-level transitions, instead of transistor-level transitions, making it extremely fast. It operates more than 1000 times as fast as equivalent transistor-level methods, and produces results that fall within a range obtained by two different industry-standard transistor-level simulators. This is an extremely useful tool for quickly comparing the energy consumption of alternative circuit architectures.

Energy consumption is calculated from a VHDL simulation of a structural DI design, using modified fundamental gates that include energy and fanout information. The output of the tool is an estimate of the energy consumed during the simulation. This value can then be divided by the number of operations in the simulation to generate the desired value of energy per operation. This method consists of two main parts: creating a VHDL energy library and transforming the structural VHDL model to use the energy library.

5.1 Energy Library Creation

First off, the energy for all set condition transitions (0→1→0) must be calculated for each fundamental gate, for a fanout of 0. In this paper the NULL Convention Logic (NCL) DI paradigm is used for testing the tool, so we calculated the energy for all transitions for each of the 27 fundamental NCL gates [6], including the necessary inverting and resettable variants. To do this, the transistor-level designs of all the threshold gates were created using Design Architect and simulated with Accusim II.

After calculating the energy for all set condition transitions for each threshold gate, the energy library was created by including this information along with each gate's functional description. Specifically, an additional energy output of type *real* was added to each gate and the gate's functional description was modified such that whenever an output transition occurs, the gate's energy per transition value is output on the additional *real* type energy output.

A generic input was also added to each gate to denote the gate's fanout. This fanout information is used to increase the energy output by a gate that switches when it has a fanout greater than 0. The gate's fanout affects its transition energy because a larger fanout means that the gate is switching a larger capacitance, and hence requires a larger amount of energy to do so. To simplify our estimation technique, we calculated the average size of a static NCL gate as 17 transistors. We then used the TH34w32 gate, which consists of 17 transistors, to test the effect that increasing a gate's fanout would have on its

energy usage. We found that increasing the fanout for any gate resulted in approximately an extra 2.0 pJ per additional gate fanout. For example, Figure 2 contains the graph of energy usage versus fanout for the TH33 gate, connected to TH34w32 gates, showing a linear trend line with a slope of 2.03. Hence, the equation for a gate's switching energy is the transition energy for the specific input combination with a fanout of 0, plus 2.0 times the gate's fanout, fo . Figure 3 shows the VHDL energy library component for the TH23 gate.

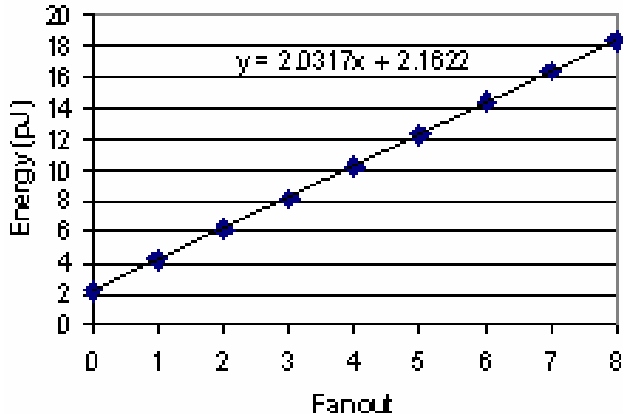


Figure 2. Energy vs. fanout for TH33 gate.

```

library ieee;
use ieee.std_logic_1164.all;
entity th23x0 is
  generic(fo: natural);
  port(energy: out real;
       a: in std_logic;
       b: in std_logic;
       c: in std_logic;
       z: out std_logic);
end th23x0;
architecture archth23x0 of th23x0 is
  signal zt: std_logic;
begin
  th23: process(a, b, c)
  begin
    if (a = '0' and b = '0' and c = '0') then
      zt <= transport '0' after tdr_th23;
    elsif (a = '1' and b = '1') then
      zt <= transport '1' after tdf_th23;
    elsif (a = '1' and c = '1') then
      zt <= transport '1' after tdf_th23;
    elsif (b = '1' and c = '1') then
      zt <= transport '1' after tdf_th23;
    end if;
  end process;
  z <= zt;
  energy_calc: process(zt)
  begin
    if zt'event and zt = '1' then
      if (a = '1' and b = '1') then
        energy <= 5.85 + (2.0 * real(fo));
      elsif (a = '1' and c = '1') then
        energy <= 5.07 + (2.0 * real(fo));
      elsif (b = '1' and c = '1') then
        energy <= 3.83 + (2.0 * real(fo));
      else

```

```

        energy <= 0.0;
      end if;
    else
      energy <= 0.0;
    end if;
  end process;
end archth23x0;

```

Figure 3. TH23 energy library component.

5.2 Energy Library Usage

After the energy library has been created, as described in the previous section, the VHDL circuit file must be modified to use the new library. As with the previously mentioned transistor-level techniques, a structural VHDL description of the circuit is required. The method then follows the steps shown in Figure 4. First, the structural VHDL description is run through Leonardo Spectrum and output as a VHDL file, so that it is in a standard format. Next, a C-language program reads this VHDL file and converts it so that it uses the new energy library. Finally, the original testbench is run on the VHDL energy file to produce the total energy consumed during the simulation.

The developed C-program executes the following steps to convert the original structural VHDL file to its energy calculating equivalent. It first calculates each gate's fanout, and includes the energy library file, which contains the component declarations for all of the energy library's components, by adding the line: `use work.energy_lib.all;`. Next, it replaces each original gate with its energy library equivalent in the circuit architecture, and declares the added *real* type energy signals and a *real* type energy accumulator signal called `energy_acc`, which is initialized to zero. The tool then inserts a process that calculates the energy consumption, which works by accumulating every transition on any gate's energy output. This process watches for a transition on any gate's energy output, and when a transition is detected, the gate's energy output is added to `energy_acc`. Finally, the program writes the VHDL energy file named `*_energy.vhd`, where `*` represents the original VHDL file name.

The VHDL energy file can then be compiled and simulated using the original VHDL testbench. After the simulation is complete, `energy_acc` contains the total energy consumed during the simulation. `energy_acc` can then be divided by the number of operations to find the *energy per operation*. For many circuits an exhaustive test can be used, where all input combinations are tested. For example, a 4-bit \times 4-bit multiplier requires 256 input vectors for an exhaustive test. This requires less than one minute to estimate the energy consumption using the method developed herein. For very large circuits a representative set of test vectors should be used in lieu of an exhaustive test.

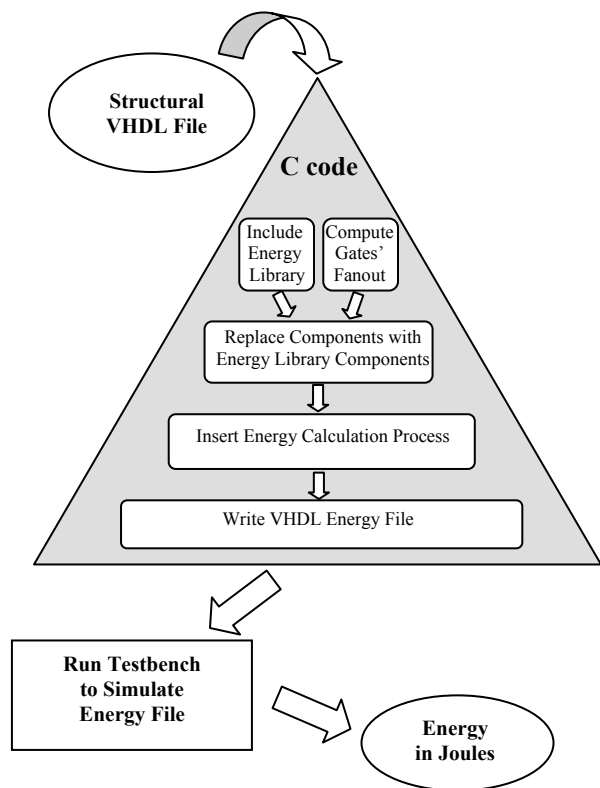


Figure 4. Energy estimation flow chart.

5.3 Full Adder Example

The estimation tool was first tested on an NCL dual-rail full adder [6], comparing the results with both the ADMS and Accusim II tools. An exhaustive test (8 input combinations) was run using all three methods, resulting in a total energy consumption of 88.0 pJ from Accusim II, 88.2 pJ using the developed VHDL-based estimation tool, and 123.2 pJ from ADMS. Note that ADMS and Accusim II are both transistor-level simulators, and that both use the same Spice model parameter file. Using the Accusim II value as the actual energy consumed, since the estimation tool's gate energy values were calculated using Accusim II, shows that the ADMS result has a 40% error, while the developed estimation tool only has a 0.23% error. Also note that the estimation tool's result falls within the range between the Accusim II and ADMS values.

6. Application to NCL Multipliers

The previous section tested the developed estimation tool on a very small design, while this section tests it using four larger designs, various 4-bit \times 4-bit unsigned multiplier architectures [12, 13], including non-pipelined

dual-rail and quad-rail versions, a full-word pipelined dual-rail version, and a bit-wise pipelined dual-rail version. Exhaustive 256-input combination simulations were run on all four designs using both the estimation tool and ADMS, with the results presented in Table I. Note that exhaustive testing of these designs is not feasible using Accusim II due to the required non-periodic signal transitioning [10]. Each ADMS simulation took slightly more than 24 hours to run on a 900 MHz Sun machine; whereas the simulations required less than 1 minute using the estimation method presented herein.

Table I. Multiplier comparison (exhaustive test).

Multiplier Architecture	Energy per Operation (pJ)	
	ADMS	Estimation Tool
Quad-Rail Non-Pipelined	529	449
Dual-Rail Non-Pipelined	736	572
Dual-Rail Bit-Wise Pipelined	1376	1040
Dual-Rail Full-Word Pipelined	1785	1318

As shown in the previous section, there is a range of acceptable results that falls between the ADMS and Accusim II values, so the purpose of these simulations isn't to compare actual energy values, but instead to compare the relative values, to show that the estimation tool can accurately estimate the relative energy requirements for the various architectures. Table I shows that both ADMS and the developed estimation tool accurately predict the relative energy usage for the four multiplier architectures. Both show the dual-rail full-word pipelined design requires the most energy per operation, followed by the dual-rail bit-wise pipelined design, the dual-rail non-pipelined design, and quad-rail non-pipelined design requiring the least amount of energy per operation. This is also intuitively correct because the non-pipelined quad-rail design would be expected to require less energy per operation compared to the non-pipelined dual-rail version since there are half as many signals transitions for quad-rail logic (i.e. two dual-rail signals transition for each corresponding quad-rail signal transition); the pipelined designs would be expected to require additional energy per operation compared to the non-pipelined version because of the additional registers and completion logic; and the full-word pipelined design would be expected to require more energy per operation than the bit-wise pipelined version because it requires additional registers and completion logic.

To further test the absolute accuracy of the developed method, it was used to calculate the energy required for only one multiplication operation on all four architectures, and the results compared with both the Accusim II and ADMS values for this operation. $1010_2 \times 0101_2$ was chosen as the test operation and was simulated on the four architectures using Accusim II, the developed

estimation tool, and ADMS. The simulations show that the estimation tool's results fall between the Accusim II and ADMS values for all four architectures, as summarized in Table II.

Table II. Multiplier comparison (one operation).

Multiplier Architecture	Energy (pJ)		
	ADMS	Estimation Tool	Accusim II
Quad-Rail Non-Pipelined	559	477	434
Dual-Rail Non-Pipelined	756	623	616
Dual-Rail Bit-Wise Pipelined	1520	1258	1240
Dual-Rail Full-Word Pipelined	1864	1755	1470

7. Conclusions

This paper developed an automated gate-level switching method for estimating the energy required for NCL DI designs. The approach is extremely useful for quickly determining how architecture changes will affect energy usage. This technique was tested on a number of multiplier architectures, and the results were shown to fall within the range of values from two different industry-standard transistor-level simulators, ADMS and Accusim II, while producing the results three orders of magnitude faster than either. Furthermore, this technique is also applicable to other gate-level DI paradigms, such as [3, 14, 15, 16, 17].

References

- [1] public.itrs.net/Files/2003ITRS/Home2003.htm (available February 16, 2005).
- [2] K. M. Fant and S. A. Brandt, "NULL Convention Logic: A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis," *International Conference on Application Specific Systems, Architectures, and Processors*, pp. 261-273, 1996.
- [3] C. L. Seitz, "System Timing," in *Introduction to VLSI Systems*, Addison-Wesley, pp. 218-262, 1980.
- [4] C. H. (Kees) van Berkel, M. Rem, and R. Saeijs, "VLSI Programming," *1988 IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 152-156, 1998.
- [5] D. E. Muller, "Asynchronous Logics and Application to Information Processing," in *Switching Theory in Space Technology*, Stanford University Press, pp. 289-297, 1963.
- [6] S. C. Smith, R. F. DeMara, J. S. Yuan, D. Ferguson, and D. Lamb, "Optimization of NULL Convention Self-Timed Circuits," *Integration, The VLSI Journal*, Vol. 37/3, pp. 135-165, August 2004.
- [7] A. Kondratyev, L. Neukom, O. Roig, A. Taubin, and K. Fant, "Checking Delay-Insensitivity: 10⁴ Gates and Beyond," *Eighth International Symposium on Asynchronous Circuits and Systems*, pp. 149-157, April 2002.
- [8] S. C. Smith, "Completion-Completeness for NULL Convention Digital Circuits Utilizing the Bit-wise Completion Strategy," *The 2003 International Conference on VLSI*, pp. 143-149, June 2003.
- [9] G. E. Sobelman and K. M. Fant, "CMOS Circuit Design of Threshold Gates with Hysteresis," *IEEE International Symposium on Circuits and Systems (II)*, pp. 61-65, 1998.
- [10] A. Singh and S. C. Smith, "Using a VHDL Testbench for Transistor-Level Simulation and Energy Calculation," *The 2005 International Conference on Computer Design*.
- [11] P. A. Beerel, C. Hsieh and S. Wadekar, "Estimation of Energy Consumption in Speed-Independent Control Circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 16/ 6, pp. 672-680, 1996.
- [12] S. C. Smith, R. F. DeMara, J. S. Yuan, M. Hagedorn, and D. Ferguson, "Delay-Insensitive Gate-Level Pipelining," *Integration, The VLSI Journal*, Vol. 30/2, pp. 103-131, 2001.
- [13] S. K. Bandapati, S. C. Smith, and M. Choi, "Design and Characterization of NULL Convention Self-Timed Multipliers," *IEEE Design and Test of Computers: Special Issue on Clockless VLSI Design*, Vol. 30/6, pp. 26-36, November-December 2003.
- [14] N. P. Singh, *A Design Methodology for Self-Timed Systems*, Master's Thesis, MIT/LCS/TR-258, Laboratory for Computer Science, MIT, 1981.
- [15] T. S. Anantharaman, "A Delay Insensitive Regular Expression Recognizer," *IEEE VLSI Technology Bulletin*, Sept. 1986.
- [16] I. David, R. Ginosar, and M. Yoeli, "An Efficient Implementation of Boolean Functions as Self-Timed Circuits," *IEEE Transactions on Computers*, Vol. 41, No. 1, pp. 2-10, 1992.
- [17] J. Sparso, J. Staunstrup, and M. Dantzer-Sorensen, "Design of Delay Insensitive Circuits using Multi-Ring Structures," *Proceedings of the European Design Automation Conference*, pp. 15-20, 1992.